AD-A080 362 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
GRAPHICAL INPUT METHODOLOGY FOR COMPUTER AIDED ANALYSIS OF CONT--ETC(U)
DEC 79 D E TROXEL
UNCLASSIFIED AFIT/GE/MA/79D-1 NL

1 OF 2
AD
A080362

AFIT/GE/MA/79D-1

(6) GRAPHICAL INPUT METHODOLOGY
FOR
COMPUTER AIDED ANALYSIS
OF
CONTROL SYSTEMS,

THESIS

(14) AFIT/GE/MA/79D-1 (10) Donald E. Troxel
Capt. USAF

(9) Masters thesis, (11) Dec 79

(12) 168

Approved for public release; distribution unlimited.

012 225

AFIT/GE/MA/79D-1

GRAPHICAL INPUT METHODOLOGY FOR COMPUTER AIDED ANALYSIS OF CONTROL SYSTEMS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Masters of Science

by
Donald E. Troxel, B.S.
Capt USAF
Graduate Electrical Engineering
December 1979

Preface

Most computer programs feel like they are designed to be operated only by a computer programmer and not by the general user. The work described in this paper is my effort to design an analysis tool that does not have the above property.

The GRAPHIC analysis system which is proposed in this paper is designed to feel natural and convenient to the user. My goal was to determine what aspects of human interfaces control the success of computer interfaces. After determining this, the GRAPHIC system was designed to meet these requirements. The resulting system is intended to be as convenient to employ by the user as normal pen and paper methods.

Donald E. Troxel

A

Contents

Contents

List of Figures

List of Tables

Abstract

This report describes the design of a graphical input language and a CAD system for specifying control system block diagrams. The goal to develop a graphical means of defining block diagrams that avoided the need to create the textual descriptions required by most existing traditional analysis packages was accomplished. Generalized requirements for a successful human interface were developed. A graphical input methodology was developed that meets these human constraints. The CAD system designed allows graphical definition of linear systems and translation of these systems into forms acceptable by traditional analysis programs.

GRAPHICAL INPUT METHODOLGY
FOR
COMPUTER AIDED ANALYSIS
OF
CONTROL SYSTEMS

I. Introduction

Background

In the guidance and control field there is the traditional need to analyze linear control systems which are represented by block diagrams. To assist in this task there are many computer aided design tools which perform various classical analyses such as root locus, frequency response and time response. The general problem with many of the computer programs for analyzing linear control systems is the complexity of the required input data. Almost all of these design tools require input that is not in the form normally used by the design engineer in his daily work. Instead of a textual description of the control system as required by current programs, the design engineer is used to representing the system with a block diagram. This diagram is a directed graph of lines and boxes, the lines representing the signal flow paths and the boxes representing the transforms that operate on the signal. The process of translating this diagram into an equivalent textual list of nodes and interconnections is time consuming and error prone.

Additionally, once the textual description is produced its complexity is such that it is difficult to detect errors and also difficult to modify. It is these problems that motivated the effort detailed in this paper.

Statement of Objectives

The objective of this effort was to design a more effective method for defining linear control systems for subsequent analysis by existing computer programs. A major goal was to employ interactive graphics as a more natural method for specifying the block diagrams of the systems to be analyzed by the computer programs.

Approach

The overall approach for this effort was divided into four phases: system definition, human interface design, system design and implementation, and design evaluation. The final section of this paper presents the conclusions that were derived from the effort and the recommendations for further work in this area.

II. System Definition

Problem Analysis

This effort was sponsored by the Control Dynamics Branch of the Air Force Flight Dynamics Laboratory (AFFDL/FGC). Since the engineers in the laboratory would become the prime users of any programs developed, the problem analysis was restricted to discussions with the laboratory personnel and a literature search.

The discussions with tne labaratory design personnel yielded the following list of deficiencies in the existing computer aided design programs such as EASY5 and TOTAL (Ref 7,12,13):

1) Input formats to existing computer programs are entirely different from any format used by the design engineers in normal daily work.

2) Current input formats are extremely complex and difficult to remember, often requiring relearning for each use.

3) The interrelationships between input elements are complicated and numerous, making detection of errors difficult.

The literature search revealed that these problems described by the laboratory personnel were common to most control system design analysis programs. While the literature showed some effective methods of interacting with the output of some tools to produce minor modifications in the analysis, no examples were found that addressed the problem of rapid and simple methods for input generation.

As a result of this analysis the overall problem was determined to be the absence of a good human engineered interface for defining control systems. Or stated in positive terms, the problem was to develop a convenient, easily remembered input method in which errors could be easily detected.

Requirements Definition

The requirements for the new design tool being implemented are derived from the general items identified in the problem analysis. The future plans for software and equipment also helped guide the development of the requirements. Each of these areas will now be analyzed in further detail.

The most significant aspect of the problem analysis is that the items identified deal almost entirely with aspects of input generation for existing computer aided analysis tools. Detailed review of the problem analysis clearly points out the requirement for the development of a new input language for describing complicated control systems. Further this new input language must be in a form that feels natural to the design engineer to allow easy input generation and verification. Thus to fill the need for a new input method a requirement for a graphical input methodology is established. A graphical method was choosen as potentially the most sucessful method since it was the means used by most design engineers when describing control systems among themselves. Also graphical methods allowed rapid error detection due to the excellent pattern recognition ability in humans.

In the laboratory there was general satisfaction with the quality and varity of existing computer aided analysis tools in terms of the analysis performed and the output data produced. Thus no requirements were produced for the generation of new mathematical analysis tools.

In the hardware and software area there is a general long ranoe plan established by the Control Dynamics Branch (AFFDL/FGC). In the hardware area this plan calls for no change in terminal equipment. This meant that the existing storage tube graphics terminals would be selected as the target interface device for human input. In the software area there are plans to combine a number of computer aided design analysis tools, including the GRAPIC system to be developed as part of this effort. This combination effort is planned to occur over several years and not involve a permanent design team. Thus the hardware requirement is to interface with the Tektronix 4000's series storage tube terminals. The software requirement is to design the new program with a modular structure to allow for future addition and combination efforts.

The complete requirements are summarized in the list that follows:

1) Develop a graphical method of generating control system definitions. This method should allow for rapid definition and modification of complex control systems for analysis.

2) Develop a method to convert the graphically defined control systems into textually defined control systems suitable for use as input to existing analysis tools such as EASY5 (Ref 7) and TOTAL (Ref 12,13).

3) Select a language and program structure that would facilitate future modifications by the Flight Dynamics Laboratory personnel.

III. Human Interface Design

This section details the steps that were taken in designing the input format and graphic language for the GRAPHIC system being developed. Since the human interface was already identified as the primary problem during the problem analysis, the approach taken here was to first make an in depth study of the factors that control the development of a successful human interface and then to design the new graphical language and other interfaces with these factors as constraints.

Human Interface Factors

The factors that effect the design of a human interface are many, complex and overlapping in scope. The method used here to analyze these factors was as follows: First, the elements of a sucessful interface were derived based on the author's experience and previous studies found in the literature (Ref 11,12,14,15,16,20). Then the aspects of human psychology and task related factors were analyzed to see how they affected the success of the interface design. Finally, the primary factors in a human interface design were considered and realistic constraints were derived to help assure production of a successful interface.

Components of a Successful Human Interface. A successful human interface for a computer program has many aspects with which it must comply. The most important of these aspects are that the interface must be friendly, easy to learn, and natural.

The nature of the interface must be friendly to the user (Ref 11,12,20). It should allow for easy detection and correction of errors. It should also allow easy changes in tasks so that the user can start one task, change to another task and then return to the first with a minimum of effort (Ref 12).

The man-machine communications format should require little training to master and should be relearnable almost immediately after a long period of non-use (Ref 16). In order for the interface to be easily relearned and retained, the language must approximate the important aspects of human to human communication. The most notable of these is that each input or output should consist of a complete thought (Ref 11,16,20). It also must have a uniform syntax. In this way, communication via the interface will feel like it is in one language instead of several languages, each with its own rules.

The interface should be natural. In this sense, it should be designed such that the user converses with the computer in terms and concepts that he normally uses when describing the steps to another human. This will cause the interface to feel "right", like a native language, instead of unconfortable, like conversing in a poorly learned foreign language.

In general, the human interface that embodies these aspects of friendliness, easiness to learn, and naturalness represents a symbiosis between the man and machine which allows the user to work as a partner with the machine to perform a task together (Ref 11). This interface should allow conversation with the machine regarding aspects of the task being performed to occur without conscious thought as to the method of expressing the individual ideas.

Human Psychological Factors Affecting Man-Machine Interfaces. Much analysis of the nature of human communication and thought has been done over the past years. These studies have developed a detailed knowledge of the physical or biological limits that control the success of human communication.

The aspects that are of greatest interest in developing a good human interface include the following: information capacity and memory characteristics, psychological closure, and psychological discomfort such as boredom, panic, frustration, or confusion. Each of these aspects will be examined in detail.

Analysis of human information transmittal capacity and short term memory have produced evidence indicating that there is a definite upper limit to the amount of data a human can absorb and retransmit. Extensive studies have shown that this limit is approximately seven items for a unidimensional measurement (Ref 14). This means that an individual can be expected to remember accurately no more than seven values of a given parameter at a time. Common examples of this would be seven lengths of a vector or seven magnitudes of a truck's speed. Further experiments have demonstrated that as the dimensionality of the data increases, the human memory capacity increases; but the accuracy of the recall for the individual parameters in each dimension decreases (Ref 14). For example, the mechanism of recognizing people involves remembering hundreds of faces; but recalling accurately the details of each face is almost impossible. This overall capacity or span of memory is also affected by the grouping or coding that is done by the individual in representing the items being remembered. In terms of numeric digits, a human could remember a binary number of about only seven consecutive binary digits, but recoding the information into decimal digits converts the capacity into seven consecutive decimal digits, the equivalent of twenty-three binary digits. In this manner it is clear that the conceptual unit of thought also affects the memory capacity (Ref 14).

In human conversation it is important that individual actions be psychologically closed or in other words be in complete thoughts. Studies have shown that transfers of information between individuals are less sucessful, in terms of correct transfer of data, if an interruption of any kind occurs during the transfer. For example, most people, after looking up a new phone number, have to look up the number again if interrupted while dialing (Ref 15,16). This is related to the fact that absolute memory capacity of a human is limited. The details of a thought are important during its expression, but once complete only the general idea of the thought need be remembered. In terms of the preceding paragraph, recoding of the information cannot be done until the thought is complete; therefore the absolute quantity of information items stored in a human memory keeps increasing until the thought is completely expressed.

Psychological discomfort in the form of boredom, panic, frustration, or confusion contributes to the failure of man-machine interfaces. As with human to human communication, man to machine communication depends on a two way exchange of information. Any interruption in this exchange results in psychological discomfort to the human. Studies have shown that it is the speed of this feedback that can make or break the sucess of a man-machine interface (Ref 15). The absolute speed of the feedback that is required varies depending on the type of input. For individual key strokes, studies show that the response must occur within 50 milliseconds in order to avoid psychological discomfort, while responses to complete thoughts can take as long as five seconds without creating ill effects (Ref 11,15).

In general, it is these human aspects of information capacity, memory, psychological closure, and psychological discomfort which must be considered as design limits in all man-machine interfaces, regardless of the task being performed, in order for the interface to be successful.

Task Related Factors Affecting Man-Machine Interfaces. There are several task related factors that affect the success of the man machine interface. These factors are task related in that the actual form of the solution to produce a good interface will vary depending on the individual combination of task and user. The primary factors are the process model, the command language, and the level of prompting.

The process model is the most important of these aspects. It is the method or the user's perception of the steps that the program takes to perform the complete task. If the process model does not match the user model, the manner in which the user conceives of the problem being solved, the interface will be a failure (Ref 11,15). The user will constantly have to translate between the steps of his thoughts, the user model, and the steps of the computer, the process model, causing needless overhead for the human. An example of this would be one in which the user considers the process to be one of "calculating" the values and "plotting" the response. The process model to support the above user model should have corresponding "calculate" and "plot" steps rather than something like "execute" and "display" steps if needless translation effort is to be reduced.

The command language must be both structured and flexible at the same time. It must be structured in two aspects. First, the language must present the user with a consistent and predictable syntax for all inputs. In this way the communications will feel like they are in one language instead of many languages. Second, the "words" of the language must match what the user normally employs to describe the steps he would perform. Specifically, the developer or programmer should not be allowed to invent new acronyms to be used as commands. At a low level this match is related to the high level matching between user and process models. In terms of flexibility, the language must support expansion and interruption. Expansion allows the user to add new "words" to the language that act as synonyms or functional groups of other "words" already defined. In this way the language can be self conforming to the user's vocabulary. Interruption as a form of flexibility refers to the ability to start a step or portion of a command and then suspend it to do some other operation. Upon completion of the second operation, the first operation would be resumed automatically from the point of suspension. Without this interruption capability, the user would have to have all inputs prepared ahead of time or be faced with starting over every time he failed to predict the need for a particular calculated value (Ref 11,12,15).

The final task related factor is the level of prompting. The level or amount of detail in the prompting must vary in two aspects. The first of these is that the detail must match the complexity of the required input. The second is that the detail must match that required by the user. To provide too much detail causes boredom for the experienced user, while to provide too little detail causes confusion for the inexperienced user. This indicates that the level of detail must be adjustable by the user to suit his taste.

It is these task related factors of process model, command language, and prompting which must be considered as design constraints if the interface is to feel natural and comfortable.

Design Constraints for a Sucessful Human Interface. It is clear from the preceding material that many factors control the acceptability of man-machine interfaces. In order to insure success, designs for interfaces must have all aspects within the constraints imposed by the human and task related factors.

These constraining factors can be best summarized in terms of complexity, vocabulary, and prompting. The complexity must be restricted to a narrow range: that is, low enough so that information can be manipulated within the capacity of the human memory, but high enough so that complete thoughts can be expressed. The vocabulary must be matched to the user. This matching must not only be in terms of the "words" of the language, but also in terms of the model of the operations being performed. Finally, the prompting must be adaptable to the user's experience level. The prompting must be such that the expert does not feel talked down to and that the novice does not feel ignored.

By careful attention to these factors it should be possible to consistantly design successful man-machine interfaces: that is, a human interface that feels natural for the particular task at hand and can be used with little if any conscious thought during the performance of the task it supports.

Interface Design

For a program such as GRAPHIC which is being developed primarily as a complicated translator of input languages from one form to another, it is clear that the design of the human interface would receive considerable attention.

In light of the discussion presented in the first half of this section, the early assumption during the requirements definition that a graphic input method was needed is well supported. For many aspects of control system specification the graphical, block diagram method truely represents the natural human method for describing linear systems. But for other aspects of the control system specification, such as specifying transfer function zeros and poles, the standard textual mode is still completely natural. For this reason the interface selected for the GRAPHIC program was designed with both a graphic interface mode and a textual interface mode. Each mode is described separately in detail. Since the User's Manual, Appendix A, explains the actual operation of the GRAPHIC program the following sections concentrate on the design and its rationale.

Graphic Interface Mode. The GRAPHIC program has six modes of operation as described in Section IV. The graphic interface mode was selected for the DRAW and GROUP modes of GRAPHIC operations. These modes are where the topology of the control system block diagram is defined.

The approach taken was to analyze what conscious actions were performed by the design engineer when sketching a block diagram on a piece of paper. From this analysis a graphic language was developed that required no more conscious thought than in the paper drawing to generate the diagram on the display screen.

On close examination of the topology specification process, it is clear that the design engineer deals with the specification problem not as how to place individual line segements but as how to order and connect the set of possible elements for the diagram. In this case the elements consist of the basic functional building blocks in a linear control system such as inputs, outputs, adders, samplers, and transform boxes. The connection of elements is specified simply by interconnecting lines. Also it appears that no conscious thought is given to the orientation of various elements or the orientation of their connections. For example for the block diagram shown in Figure 1, there is no conscious differentiation between transform one and transform two due solely to the fact that one has its input on the left while the other has its input on the right. Similarly in Figure 2 there is no conscious differentiation between adder

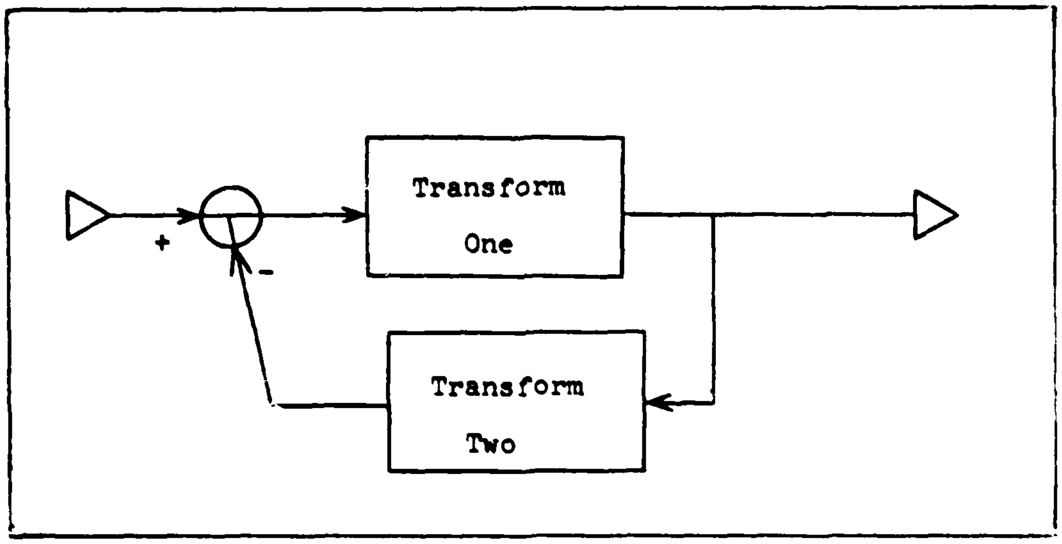


Figure 1. Block Diagram Topology Example

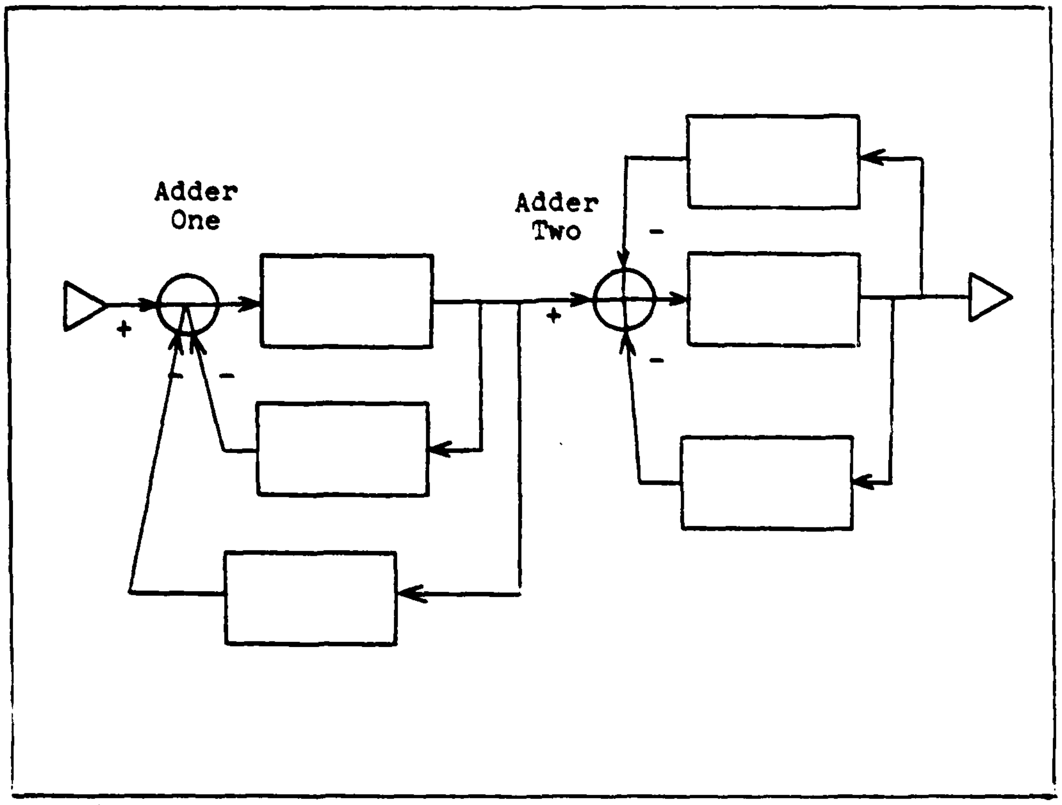


Figure 2. Block Diagram Topology Example

one and adder two due solely to the fact that one has two inputs from below while the other has one input from below and one from above.

Clearly, in order to retain natural feeling to the interface it will be necessary to handle all subconscious aspects of block diagram drawing automatically within the computer. In this way, the visible interface will require no more conscious thought for use than needed for sketching on paper. Specifically, this means that the computer will have to keep track of orientation and input output pairing of elements within the diagram. To do this the concept of a current working location (CWL) and drawing direction vector are used. The current working location is the XY position that is the location of the output for the most recently drawn symbol. The drawing direction vector is a vector pointing in the same direction that the last line element was drawn in. The visible effect of the CWL and drawing direction vector are that as elements are added to the diagram they appear connected to the previous element and in the proper orientation automatically. The drawing convention is that all elements are specified in the order corresponding to the direction of signal flow. This means that the user would define the diagram starting at an input and working toward any output. This convention allows automatic generation of clarifying marks such as arrowheads to help indicate the input side of elements in the block diagram.

For the paper sketch example the user consciously decided only what element to draw next and where to draw it. With this in mind the interface requires only the equivalent information, the code letter for the element desired and the crosshair position for the element location. The individual code letters were chosen to be the first letter of the elements name as shown in Table I. In order to retain as much of the natural feeling as possible and avoid conscious translation effort, the names of the elements were picked to be the names most often used by actual design engineers.

In addition to the commands depicted in Table I several other commands are needed for complete generality. These commands are required to delete, modify, and review the elements in the block diagram and are described in Table II. For all of these commands the first letter of the command name was used as the command code unless there was a conflict.

This form of interface produced the most convenient and rapid method available for defining block diagram topologies. It was one in which actions could be accomplished based on one character inputs from the user.

The next most important aspect of the graphics interface is the layout and use of the display screen. Since the terminal was a storage tube connected to a timesharing network certain aspects of the interface had to consider the constraints that this imposed. The particular thing to avoid was the need to erase some small item in the display and

Table I

Element Producing Commands in Draw Mode

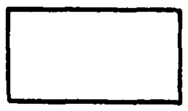
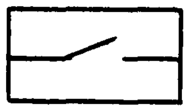
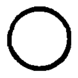
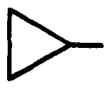
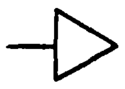
Name	Code	Representation
Box	B	
Sampler	S	
Adder	A	
Input	I	
Output	O	
Line	L	

Table II

Non-Element Producing Commands in Draw Mode

Name	Code
Move	M
Connect	X
Delete	D
End	E
Working Location	W
Center	C
Redraw	R

retransmit all the elements that were to remain visible. This erase and retransmittal of the data was particularly undesirable due to the low data rate of the timesharing network, 30 to 120 characters per second.

For the above reasons the display face was organized as shown in Figure 3. The display was organized into four major areas: prompt, work, miniature, and text areas. The graphic prompt area lists all of the commands defined in the graphic language. Each command is shown with its corresponding graphic symbol or descriptive mnemonic (Figure 4). This approach, displaying the entire list of commands,

was taken since there are fifteen different command codes. As mentioned earlier a human can only be expected to accurately remember seven or less items in a set such as the command code set. The text message area is used by the program to post error and status messages. The messages in the text area appear one after the another on each line using every character position before starting the next line. This feature reduces the frequency of display erasures due to insufficient text message space for the next message to be posted. The work area is where all of the drawing is done to create or modify the block diagram topology. The detailed explanations of the drawing commands are given in the User's Manual, Appendix A. In addition to the commands to create the diagram elements, there are also commands to position the entire diagram within the work area. For this purpose the drawing can be thought of as being on a large sheet of paper

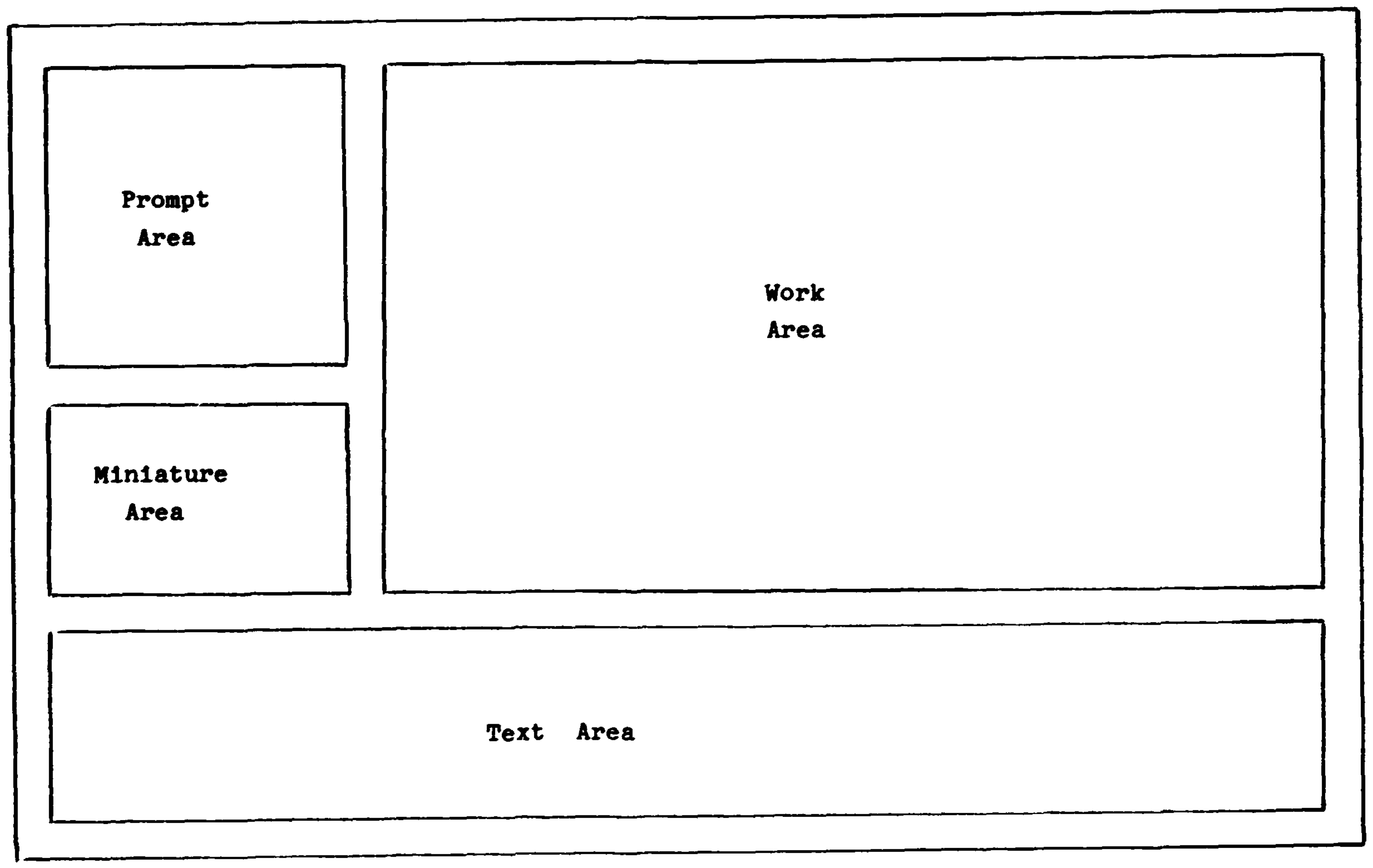


Figure 3. Terminal Display Face Organization

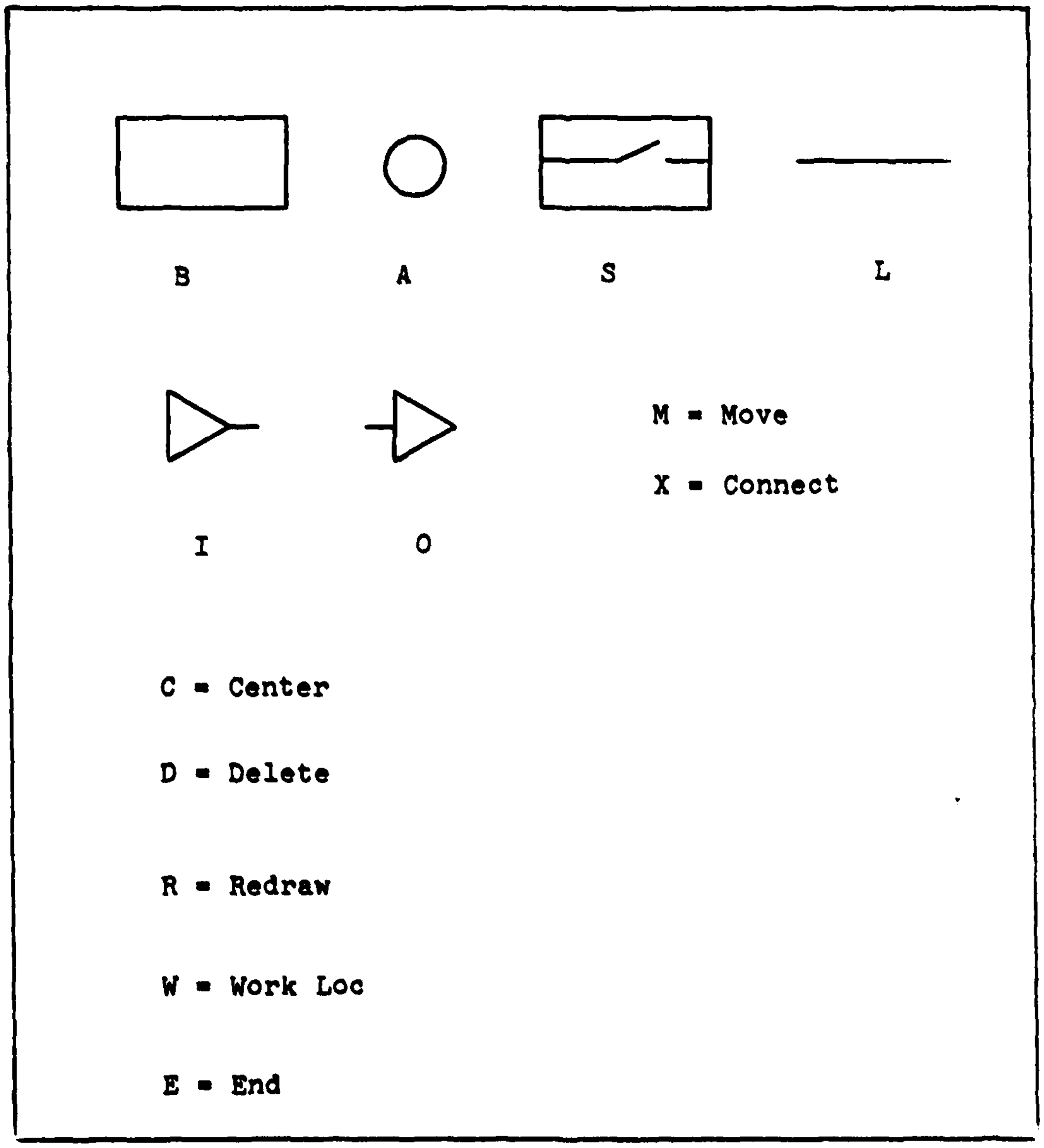


Figure 4. Contents of Prompt Area

which is viewed through a window called the work area. The CENTER command allows the user to direct which element in the drawing is to be centered in the middle of the work area. The system keeps track of the total diagram size and whenever any of the elements disappear outside the window edge a minature display of the entire diagram without the element labels is produced. The production of this minature can be inhibited by the user if desired. This approach was taken to allow the experienced user to avoid the delay caused by the transmission time needed to produce the minature.

Based on the analysis of the system requirements given in Section II and the constraints on designs for successful human interfaces, this graphic interface was implemented to allow efficient and natural definition of block diagram topologies.

Textual Interface Mode. The textual interface mode is used for all aspects of input and output to the GRAPHIC program that are not directly associated with generation of the block diagram topology. The major areas of use for this interface mode are in the PARAMETER and REDUCE features of the of the GRAPHIC program. In the PARAMETER mode the transfer functions are specified while in the REDUCE mode the form of the reduction is specified.

The textual mode is used when there is no graphic data present on the screen. It is used to communicate with the user in a teletype style that is normally associated with computer communications. Each group of inputs or outputs uses a new line whether or not the previous line was full. Although this method wastes space on the face of the storage tube display and requires more frequent erasures it was still determined to be more beneficial than the text communication method used in the graphic interface. The main reason was that it eliminates the internal bookkeeping and calculations required to post messages in the graphic mode. This allows for the most rapid response available and therefore helps prevent user frustration. The second reason is that it allows for longer prompting messages. Longer messages are possible since there is no concern as to how often the display screen fills up and has to be erased. Unlike operations in the graphic interface mode, where large amounts of data, specifically the block diagram, had to be redrawn after each erase, in the text mode the old data corresponds to already answered questions and can be discarded with the erase. The data to be input and output during the PARAMETER and REDUCE modes of GRAPHIC is primarly textual in nature, consisting mostly of numeric coefficients and element labels.

Comparison to Requirements and Constraints. This combination of graphic and textual input modes was implemented for the GRAPHIC program as the most likely design for the human interface considering the system requirements stated in Section II and the design constraints stated in the first half of this section.

The graphic interface mode has a command set that is rich enough to allow definition and modification of block diagrams representing arbitrary topologies of linear systems. The complexity of the user inputs is low, being in the general case the specification of a location with the crosshair and a command with a code letter. The vocabulary of the graphic language is matched to the user since the mnemonics for the command names are taken directly from the most commonly used term employed by the actual design engineers who will be the primary users. Finally, the prompting display of user options is constantly presented in the graphic prompt area (see Figure 4). This approach frees the user from memorizing the command set. The production of the minature, another form of prompting, can be disabled depending on the user's preference.

The textual interface mode allows for complete specification of the required cofficient data and reduction parameters for output translation. Its complexity and vocabulary are ideally suited for the large amounts of numeric data that must be input in this mode. The elimination of the graphic data from the screen allows more rapid and complete prompting to be accomplished.

This combination of interfaces was implemented as being the best choice for the combined requirements. It is designed to allow rapid definition of control systems in the form of arbitary block diagrams. The ability of this interface design to perform this function while retaining a natural communication feeling in the user will be discussed in Section V.

IV. System Design and Implementation

The GRAPHIC program was designed to be implemented using a highly modular structure at all levels (Ref 10,22). This modularization was designed in for several reasons. First, it allowed nearly independent implementation of the major functions. Second, adding of new features in the future would not require dissecting the entire package but simply connecting a new module. Finally, grouping the code into modules makes it easier to understand. This last item is relatively important since the author would not be the maintainer of the developed system.

Within the code, two additional features were employed to increase readability and understandability on a system wide basis. Whenever possible parameters passed between routines either as formal parameters or as common data structures were given the same alphanumeric identifier in all routines. Also the use of numeric constants is avoided in preference for the use of variables that are treated as constants. Both of these features help to clarify where the value being used comes from and also provide a better indication of when the value is used.

Program Overview

The GRAPHIC program is written entirely in FORTRAN for the Control Data Corporation (CDC) CYBER series computer (Ref 4,8). It is designed to run as an INTERCOM job under the NOS/BE operating system (Ref 1,3,5,9). For this reason the code is overlayed to reduce its memory requirement to under 65,000 base eignt words at any given instant. The GRAPHIC program is designed to be used only at a Tektronix 4000 series terminal.

As shown in Figure 5 the basic overlay structure of GRAPHIC consists of one main overlay and six primary overlays. Each of the six primary overlays corresponds to one of the six major features or modes of operation in GRAPHIC.

The GRAPHIC program uses support routines from three specialized libraries to obtain services that are not a part of the normal CDC FORTRAN environment. These libraries are TEKLIB, NOSLIB, and CORE. The TEKLIB library is a set of routines that provide the low level drivers to interface directly to the user's Tektronix 4000 series terminal (Ref 18). The NOSLIB library provides file manipulation capabilities for cataloging files (Ref 2). The CORE library is a complete set of subroutines implementing a general purpose three dimensional graphics system. This three dimensional system is modeled after the proposed standard for graphics interfaces produced by the Association for Computing Machinery (ACM) Special Interest Group for Graphics (SIGGRAPH) (Ref 17). The TEKLIB and NOSLIB libraries are maintained by the ASD computer center at Wright- Patterson AFB, Ohio (Ref 6). The CORE library was developed by a group of Masters students including the author at the Air

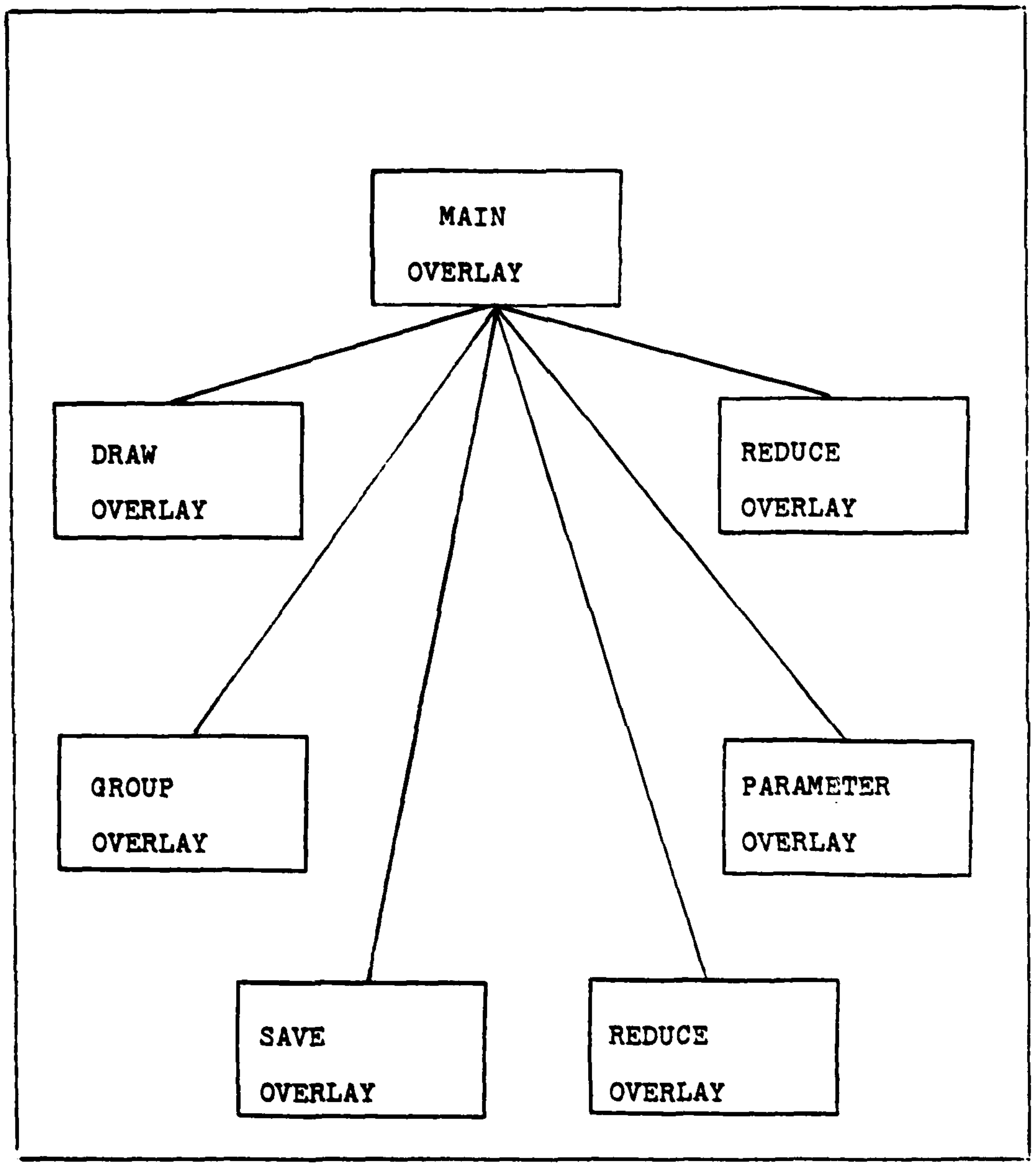


Figure 5. Basic Overlay Structure of GRAPHIC

Force Institute of Technology as part of a class project (Ref 19) and is further described in Appendix C.

Main Overlay. The main overlay is resident in memory at all times when the GRAPHIC program is in operation. As depicted in Figure 6, the main overlay handles the initialization of global variables and controls the execution of each of the six primary overlays.

Besides initializing all of the global variables upon start up, the main overlay acts as a storage area for shared data. Since this overlay is the only overlay that is resident in memory continuously, any data that is shared or passed between the primary overlays must be stored in the main overlay.

Once the global variables have been initialized the only active function the main overlay has is to select which primary overlay to execute next. This is done based upon the user input and results in execution of primary overlay DRAW, GROUP, PARAMETER, REDUCE, SAVE, or RESTORE. After the primary overlay completes execution, control is returned to the main overlay and another primary overlay is selected for execution.

Draw Overlay. The DRAW overlay is responsible for accepting the block diagram definition from the user and storing it in the graphic database. As depicted in Figure 7, it has a separate routine for each major operation that can be performed during the block diagram definition.

The graphical interface mode is the only form of

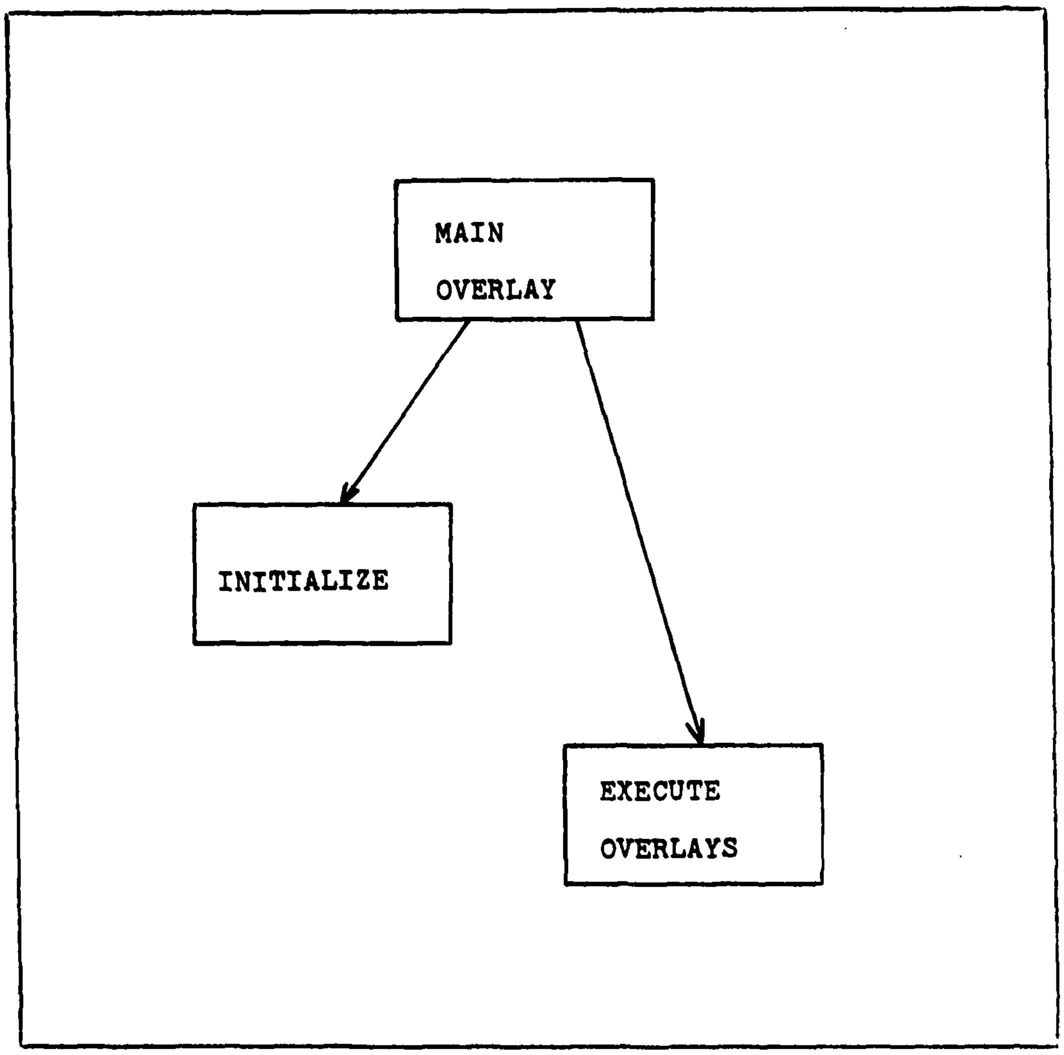


Figure 6. Main Overlay

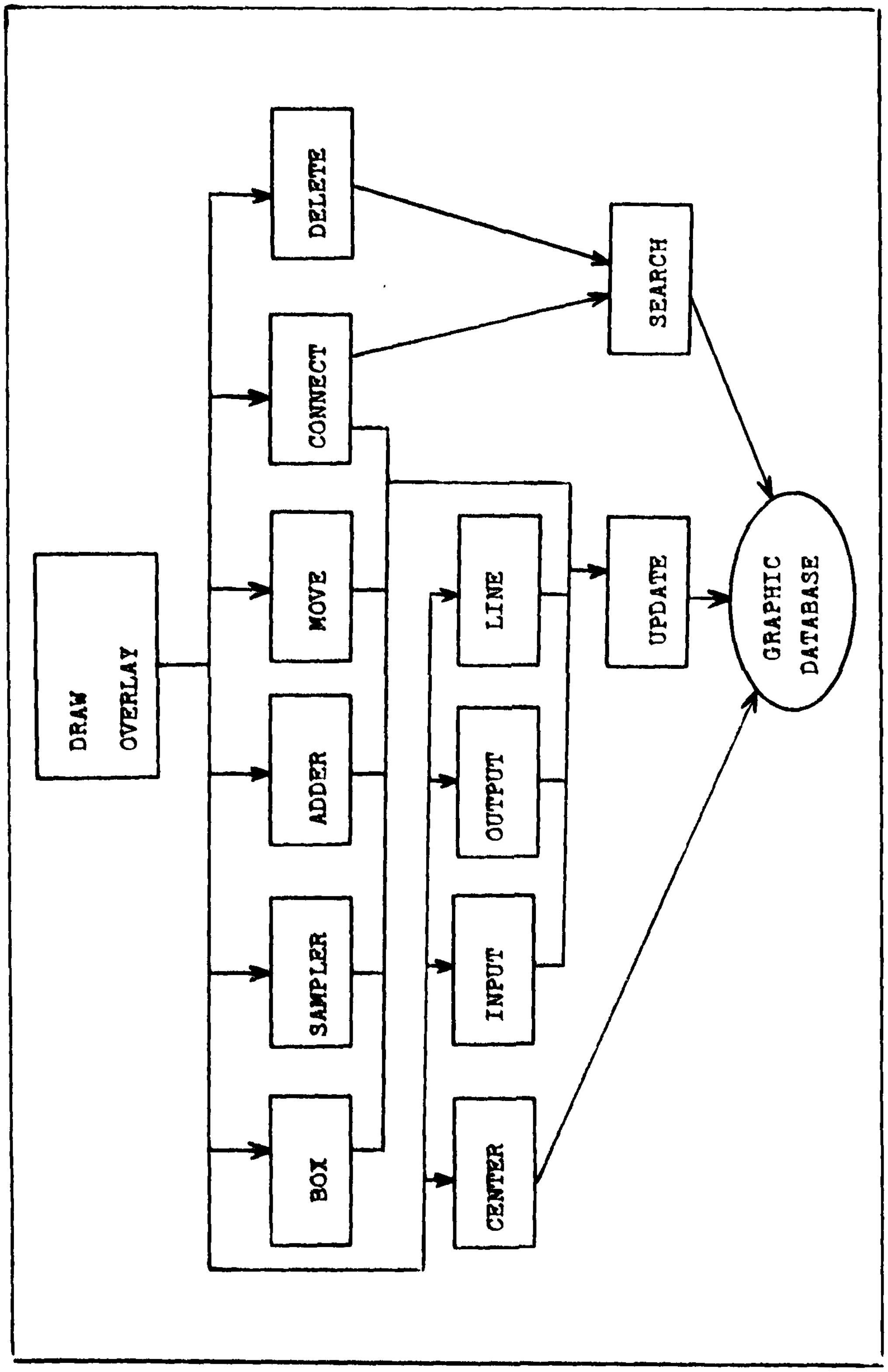


Figure 7. DRAW Overlay

communication with the user while running under this overlay to define a diagram. This means in the general case that all inputs are restricted to a one letter command code and in some cases an additional XY location given by the crosshair position. Table III contains a complete list of the commands available and their meaning.

The DRAW overlay operates as a ten way switch: requesting a command from the user, performing the operation indicated, and then requesting the next command. Each time the program is ready for a new command the crosshair appears on the screen. While the system is busy processing the input the crosshair is removed from the screen. This feature provides the user with positive feedback as to whether an operation is in progress or not. The switch structure of the overlay allows new features or commands to be added to the graphic interface simply by modifying the section of code that forms the input command filter.

As each element in the block diagram is defined a corresponding entry is established in the graphic database. For all elements this is done by the UPDATE routine. This design was chosen to centralize access to the graphic database. In this way any changes to the internal structure of the graphic database could be hidden from the other routines and handled entirely within the UPDATE routine.

The graphics presented on the face of the display are produced using the three dimensional drawing support routines in the CORE library. Although all items are represented as

Table III

DRAW Mode Commands

Command	Code
Box	B
Sampler	S
Adder	A
Input	I
Output	O
Line	L
Move	M
Connect	X
Delete	D
Working Location	W
Center	C
Redraw	R

two dimensional sketches, a three dimensional support package was used to allow easy expansion into the third dimension should a meaningful way of representing control system block diagrams in three dimensions be developed in the future.

Actual manipulation of the graphical data is controlled using the following concept of separate coordinate systems. All data in the graphic database such as specific XY points for each symbol is treated as locations in an infinite User Coordinate System (UCS). The display terminal face is treated as a separate Device Coordinate System (DCS). In this manner data in the graphics database completely describes the block diagram in UCS. The production of a picture of the diagram on the display face is just a mapping of the points within a window in UCS into a viewport in DCS space. This concept is quite powerful and, with the addition of clipping to the window boundaries, allows the symbols in the work area, miniature area, and prompt area to be produced entirely by the same software with only changes in the mathematical mapping. This assures unchanging proportions between the symbols and the different areas in which they appear since there is only one set of vector commands for each graphic symbol type: box, sampler, adder, input, output, and line.

Group Overlay. The GROUP overlay is responsible for creating collections of elements as specified by the user's input. Once created, the collections can be used in the DRAW mode by referring to them with one command code for each individual collection. The specific details on how this is done are in Appendix A.

The GROUP overlay is essentially a specialized form of the DRAW and PARAMETER modes. It is structured as depicted in Figure 8. The major sections of code in this overlay are just an encapsulated copys of the DRAW and PARAMETER overlay code. The other section allows the existing groups of elements to be scaned and any new groups to be generated.

Parameter Overlay. The PARAMETER overlay is responsible for acquiring the numeric data to complete the mathematical definition of the diagram defined by the DRAW overlay. Its structure is depicted in Figure 9. This overlay allows the user to specify the numeric values of the polynomials that form the transfer functions for each box element in the diagram. It also allows specification of the sampling period for every sampler in the diagram.

The program is set up to cycle through the list of box and sampler elements requesting the needed parameters or to allow the individual parameters to be set one at a time based on the user input. The polynomials that form the transfer function can be specified by either a list of coefficients or a list of roots on an individual polynomial basis. Additionally, there are key words and mnemonics to allow skipping over unchanged roots when redefining parameters. The complete details of how this would be accomplished are in the User's Manual, Appendix A.

Regardless of whether the polynomial parameters are

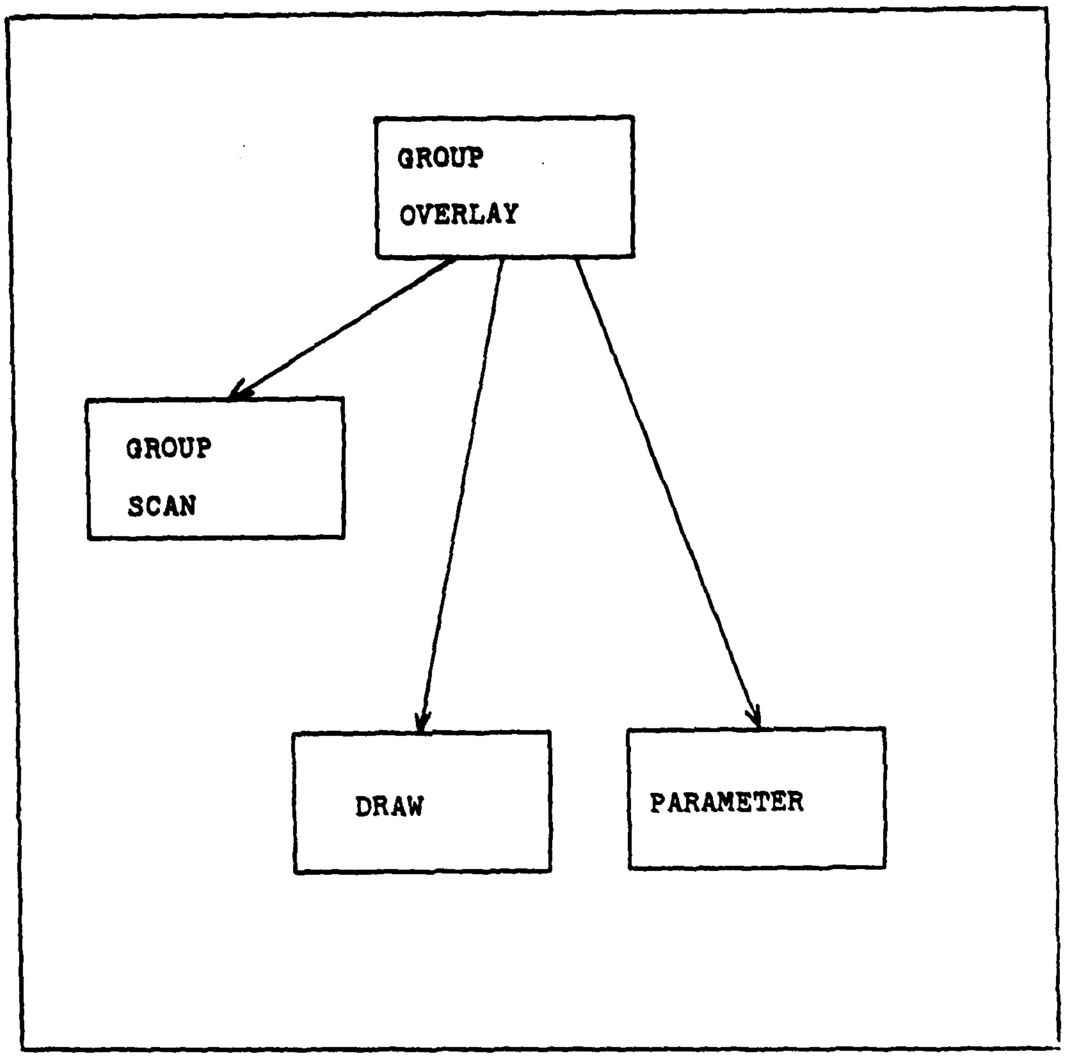


Figure 8. GROUP Overlay

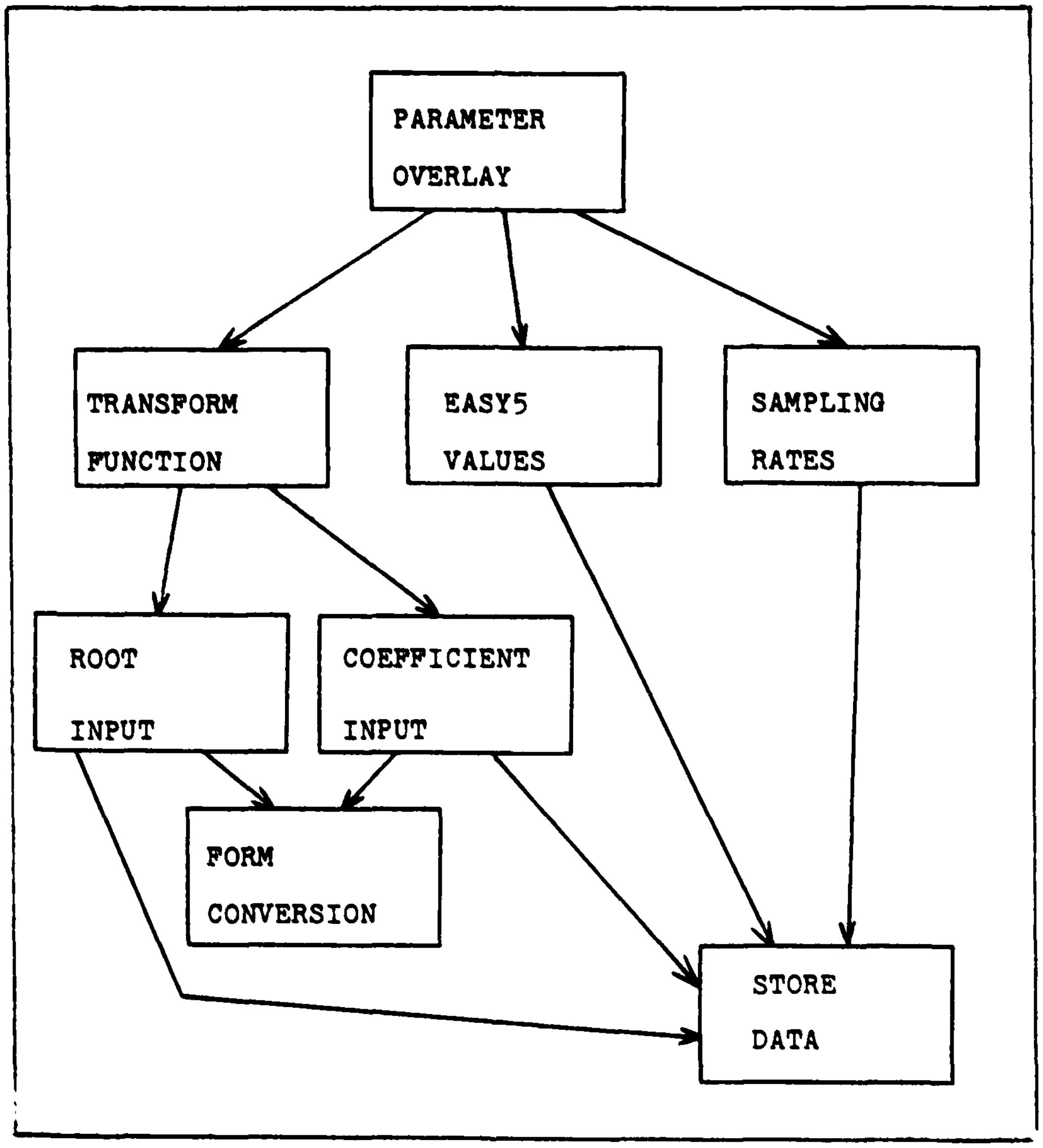


Figure 9. PARAMETER Overlay

entered as coefficients or roots, they are stored in both forms in the database. This approach was taken to allow direct compatibility with the storage format in use by TOTAL (Ref 12). Additionally, the routines used by TOTAL to convert between coefficient and root form can be employed in GRAPHIC without modification.

Reduce Overlay. The REDUCE overlay is responsible for reducing the data stored in both the graphic and numeric databases. The structure of this overlay is depicted in Figure 10. REDUCE operates in two general modes. In one mode it produces a file of card images for use as input to the EASY5 program (Ref 7). In the other mode it calculates the equivalent transfer function for a subset of the block diagram and stores the answer for further use in interactive processing.

Regardless of which mode is in effect, the program first makes any temporary changes in the block diagram topology that the user requests and then acquires the points on the diagram between which the equivalent transfer function is to be calculated. Then the graphic database is analyzed and compressed to form the signal flow database. The FLOW subroutine performs this work which amounts to discarding the graphical position data to create a signal flow graph that retains only the mathematical connectivity information. The signal flow database is small enough to fit entirely in memory. From this point the processing diverges depending on the mode of operation desired.

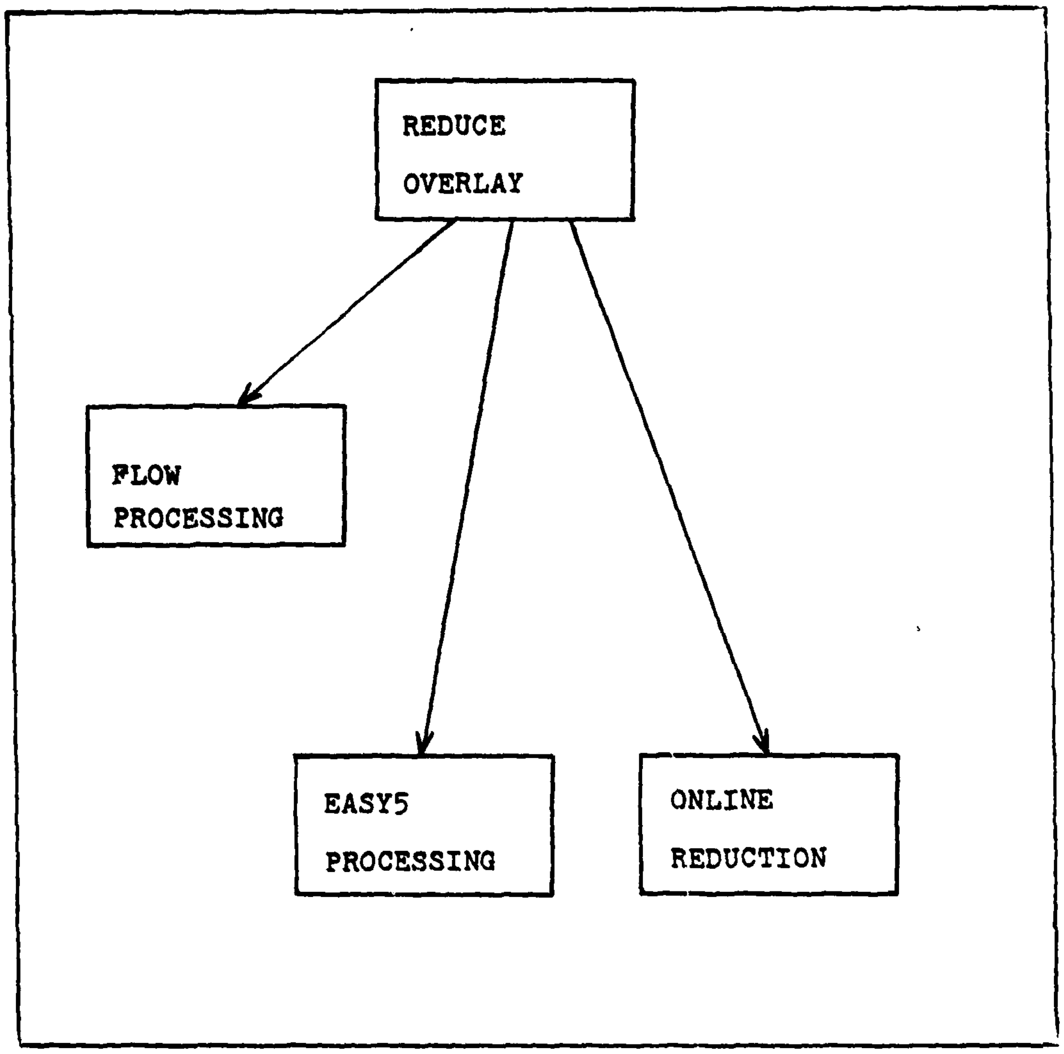


Figure 10. REDUCE Overlay

NOS/BE operating system (Ref 1,3,5,9). For this reason the

29

If the EASY5 mode is used the signal flow database and the numeric database are expanded and converted to an equivalent textual description describing the block diagram topology and polynomial roots. This operation of mapping from the GRAPHIC representation into the EASY5 representation is somewhat complicated since it is a one to many mapping. This is because GRAPHIC allows transfer functions as large as order 50 while EASY5 must cascade functions of order 2 or less. Additionally, while GRAPHIC has an arbitrary element, the transform box, for any transfer function, EASY5 has many different elements depending on the combination of numerator and denominator polynomial orders. Once produced, the text file describing the block diagram is available for use without further modification as the complete input file for EASY5.

If the interactive mode is selected, the signal flow and numeric databases are analyzed directly and the equivalent transfer function is calculated and stored as a special entry in the numeric database. This allows the result to be used in later diagrams and analyses. The method used to find this transfer function is a computerized implementation of Mason's reduction method developed by Young (Ref 21).

In this overlay the division of tasks in operation is such that all information shared between subroutines is stored in the signal flow database. This allows the routines for FLOW, EASY5 mode, and interactive mode processing to be implemented as secondary overlays and therefore have their code areas share the same memory space but operate at different times.

Save Overlay. The SAVE overlay is responsible for permanently storing the GRAPHIC databases so that they can be reused at a later time. This feature eliminates the need to redefine the block diagram everytime the program is activated. The structure of this overlay is depicted in Figure 11.

The three step process used to save the graphic and numeric databases consists of the following items. First, the desired filename is obtained from the user. Second, the databases are converted from their ramdom mass storage form to a more compact sequential form. Finally, the sequential file is passed to the operating system for cataloging as a permanent file. This method of storing the database is similiar to that used by the TOTAL program (Ref 12).

Restore Overlay. The RESTORE overlay is responsible for rebuilding the databases from information in a permanent file established by a previous SAVE command. The structure of this overlay is depicted in Figure 12 and closely parallels the structure of the SAVE overlay.

The RESTORE overlay performs a three step process that is the logical inverse of that performed by the SAVE overlay. First, it obtains the name of the file containing the stored data. Second, it requests the operating system to attach the specified file to this program. This operation uses the routines in the NOSLIB library (Ref 2). Finally,

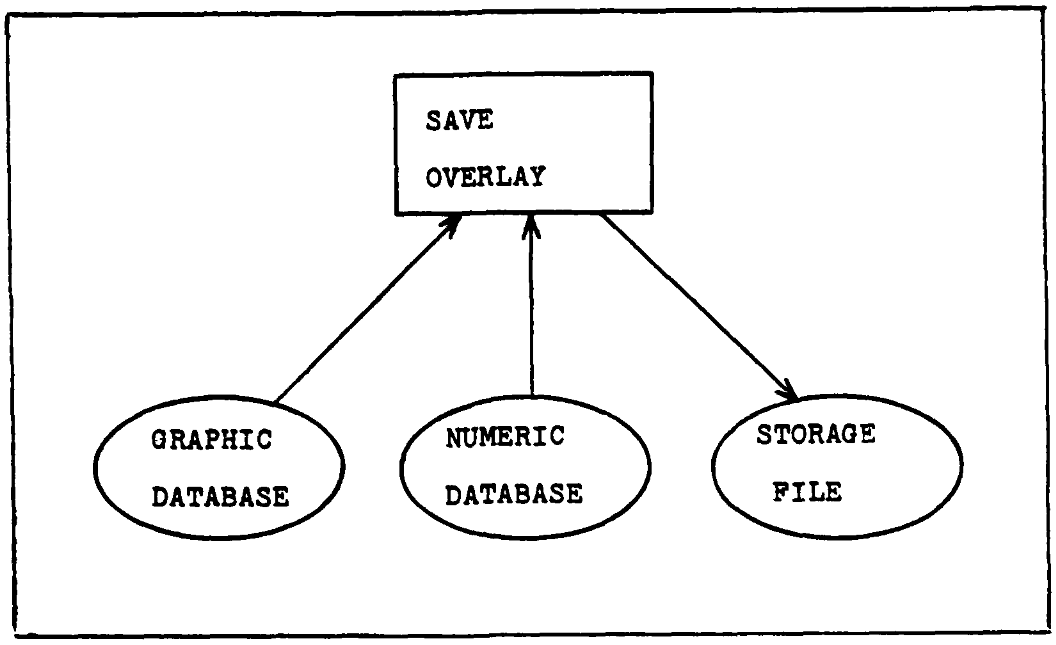


Figure 11. SAVE Overlay

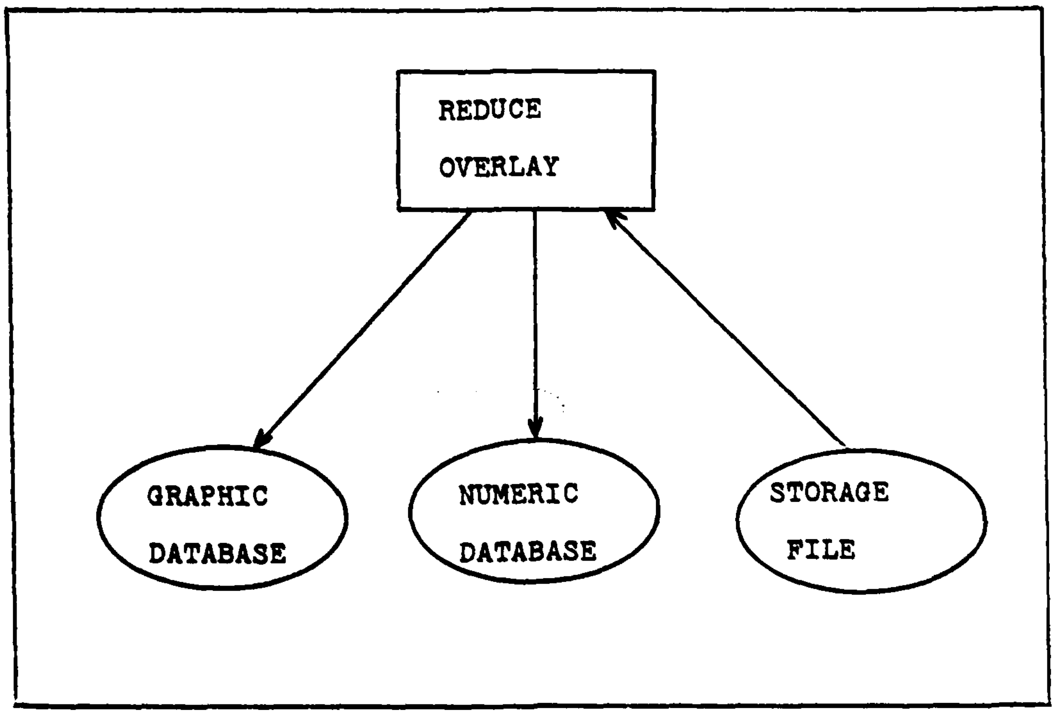


Figure 12. RESTORE Overlay

it reads the data from the file and recreates the two separate ramdom access mass storage databases of graphic and numeric data.

Data Structures

Large amounts of data are received and stored by the program in the process of accepting the block diagram definition from the user. That data is functionally categorized and divided into three general data structures or databases. As mentioned in the preceding parts of this section, these data structures are the graphic database, the numeric database, and the signal flow database.

The graphic and numeric databases are shared among all five of the primary overlays. The data in these two databases is derived directly from user inputs. The signal flow database is used only by the REDUCE overlay. The data in the signal flow database is derived from the content of the graphic database.

Graphic Database. The graphic database is generated from the commands input by the user during the DRAW mode of operation. It contains all the information needed to reproduce the drawing of the block diagram on the terminal display screen. Additionally, the information on the diagram topology, specifically what symbols connect to each other, is also contained in this database.

As the user enters commands to define elements of the block diagram, corresponding records are created in the database. One record is created for each box, sampler, adder, input, output, and line element defined by the user. A connection command generates a record only if a diamond connection symbol was produced. Otherwise the connection command results only in a change in the topology information.

The records of the database are structured as shown in Table IV. Each record contains all the necessary information to define the representation of the element and its connectivity to other elements in the drawing. In this regard the record has entries for the symbol type, location, and orientation as well as the element label. Additionally, the record contains pointers to the records for the elements which are connected to it in the diagram. In this way the database records form a doubly linked list indicating the signal flow paths. If the record is for a box or sampler element it also contains pointers into the numeric database to the location of its describing paramaters.

To make it easy to access the database, linked lists of the records are maintained for each element type. A separate linked list is maintained for each set of box, sampler, adder, input, output, line, and connection element records. Additionally, the maximum size or extent of the block diagram is maintained as a separate parameter for use in controlling the generation of the optional miniature diagram. The collection of linked lists and diagram size data are treated as two nonstandard records in the graphic database. The structure of these records is given in Table V.

Table IV

Graphic Database Record Format

Element	Type	Use
1	ASCII	Command code for graphic symbol type.
2	Real	X value of current working location.
3	Real	Y value of current working location.
4	Real	X value of end point for line symbol.
5	Real	Y value of end point for line symbol.
6	ASCII	System generated visible label.
7		Unused.
8	ASCII	User supplied label.
9	ASCII	Flag for S or Z domain.
10		Unused.
11	Integer	Pointer to numerator polynominal in Numeric Dataase.
12	Integer	Pointer to denominator polynominal in Numeric Database.
13	Integer	Pointer to record for output element.
14	Integer	Pointer to record for output element.
15	Integer	Pointer to record for input element.
16	ASCII	Sign of input from element entry 15.
17	Integer	Pointer to record for input element.
18	ASCII	Sign of input from element entry 17.
19	Integer	Pointer to record for input element.
20	ASCII	Sign of input from element entry 19.
21	Real	Direction vector X component.
22	Real	Direction vector Y component.
23	Real	Minimum X limit of element.
24	Real	Maximum X limit of element.
25	Real	Minimum Y limit of element.
26	Real	Maximum Y limit of element.
27	Real	Line direction vector X component.
28	Real	X location of element label.
29	Real	Y location of element label.
30	Integer	Element label lenght.
31		Unused.
32		Unused.
33		Unused.
34	Real	Line direction vector Y component.

Table V

Specialized Records in Graphic Database

a) Linked List (entry 1001)

Word	Use
1-1000	Array of 6 interwoven linked lists.
1001	Pointer to start of box list.
1002	Pointer to end of box list.
1003	Pointer to start of sampler list.
1004	Pointer to end of sampler list.
1005	Pointer to start of adder list.
1006	Pointer to end of adder list.
1007	Pointer to start of input list.
1008	Pointer to end of input list.
1009	Pointer to start of output list.
1010	Pointer to end of output list.
1011	Pointer to start of line list.
1012	Pointer to end of line list.
1013	Pointer to start of connect list.
1014	Pointer to end of connect list.
1015	Pointer to first free location in array of linked lists.
1016	Count of box entries.
1017	Count of sampler entries.
1018	Count of adder entries.
1019	Count of input entries.
1020	Count of output entries.
1021	Count of line entries.
1022	Count of connect entries.

b) Drawing Extent (entry 1002)

Word	Use
1	Left edge of diagram in UCS.
2	Right edge of diagram in UCS.
3	Bottom edge of diagram in UCS.
4	Top edge of diagram in UCS.

Note: * UCS = User Coordinate System. (Units the drawing is referenced to in the database.)

Because of its size, no attempt is made to keep the graphic database in memory. Instead, the database is organized as a random access mass storage file. Although this approach introduces additional overhead for each access to the database the delay is not significant. The random access method was chosen since most of the time only one element record is of interest while processing a given command. It is only for the DELETE and CONNECT commands that the database needs to be completely searched. The linked lists allow the PARAMETER mode to selectively access only the box and sampler element records and avoid the need to read the rest of the records. The only other time all the database records are read is when the diagram is redisplayed after the CRT screen is erased or when the signal flow database is generated.

Numeric Database. The numeric database contains all information necessary to define the mathematical properties of the arbitrary transform boxes and samplers in the graphic drawing. It contains one record for each transform and one combined record for all sampling rates.

The individual transform records each contain the mathematical description of the transfer function stored as a ratio of polynomials. The functions are stored in both coefficient and root form as shown in Table VI. The format chosen for this data is identical to the storage format used by TOTAL in order to increase future compatibility between the two programs.

Table VI

Numeric Database Record Format

Element	Type	Use
1-51	Real	Coefficients of numerator polynominal.
52-102	Real	Coefficients of denominator polynominal.
103-152	Real	Real part of numerator polynominial roots.
153-202	Real	Complex part of numerator polynominal roots.
203-252	Real	Real part of denominator polynominal roots.
253-302	Real	Complex part of denominator polynominal roots.
303	Integer	Order of numerator polynominal.
304	Integer	Order of denominator polynominal.
305	Real	Gain of transfer function.
306	Real	Gain of numerator polynominal.
307	Real	Gain of denominator polynominal.

The sampling rates of all samplers in the system are stored as one record at the end of the database.

The database is created during the "PARAMETER" mode of the program and resides on a mass storage device. It can be reused in later invocations of the program, but only with its corresponding graphic database.

Signal Flow Database. The signal flow database is created during the REDUCE mode of operation. It is generated from the data in the graphic database and has pointers into the numeric database.

This database represents the signal flow graph of the diagram defined during the DRAW mode. As such it contains only information regarding the diagram topology or connectivity. All information from the graphic database regarding labeling or the absolute positioning and orientation of elements is discarded. The records of the signal flow database occur in three different formats. The exact content of each record format is shown in Table VII.

The signal flow diagram, unlike the graphic and numeric databases, is maintained entirely in memory. This is done to allow easy and rapid access to all records during the reduction process. This access is needed since the process of translating the diagram into EASY5 format as well as the process of finding the equivalent transfer function both require repeated accesses to all records of the signal flow database.

Table VII

Signal Flow Database Record Format

a) Format for Adder Elements:

Entry	Type	Use
1	Integer	Key of original record in graphic database.
2	ASCII	Symbol type indicator.
3	Integer	Pointer to output element record.
4	Integer	Pointer to input element record.
5	Integer	Pointer to input element record.
6	Integer	Pointer to input element record.
7	ASCII	EASY5 label assigned to this element.
8	Integer	Pointer to output continuation record.

b) Format for Output Connection Records:

Entry	Type	Use
1	Integer	Unused.
2	ASCII	Flag indicating output connection record.
3	Integer	Pointer to output element record.
4	Integer	Pointer to output element record.
5	Integer	Pointer to output element record.
6	Integer	Pointer to output element record.
7	Integer	Pointer to input element record.
8	Integer	Pointer to next connection record.

c) Format for Box, Sampler, Input, and Output Elements:

Entry	Type	Use
1	Integer	Key of original record in graphic database.
2	ASCII	Symbol type indicator.
3	Integer	Pointer to output element record.
4	Integer	Pointer to output element record.
5	Integer	Pointer to output connection record.
6	Integer	Pointer to input element record.
7	ASCII	EASY5 label assigned to this element.
8	Integer	Pointer to parameters in numeric database.

V. Design Evaluation

The design of the GRAPHIC system and its interfaces as discussed in the two preceding chapters is fully documented in the form of a User's Manual which appears as Appendix A to this paper. Although the GRAPHIC system was designed with the interface constraints discussed in chapter three in mind, it was necessary to verify the actual feasibility of the proposed design. For this reason a subset of the total system was implemented. The area of specific interest was the suitability of the proposed graphic input language.

System Implementation.

The system implemented to test the graphical interface with the user contained only selected features from each of the modes of operation of GRAPHIC. In the DRAW mode, the entire set of basic commands was implemented. Only the user group feature, which allowed formation of collections of symbols, was omitted. In the PARAMETER mode, the coefficient and root forms of input were implemented. The input for sampler periods was omitted. In the REDUCE mode, the EASY5 translation for continuous systems was implemented. The digital or sampled systems and the online reduction features were omitted.

Although some of the features were omitted for the test, the capacities of the internal tables were maintained at their full size estimates. This was done to obtain realistic estimates of the amount of memory space needed to implement the complete version of GRAPHIC as defined in Appendix A. In this regard the graphic database was implemented to allow 1000 elements (records) and the signal flow database was implemented to handle 500 nodes (records). These sizes are proportional to each other since the signal flow database contains approximately one entry for every record in the graphic database that is not a line or connection element record. The items omitted from this system implementation, as discussed earlier, did not effect the memory size estimates since they were to be implemented as parallel or secondary overlays and therefore share space with the modules implemented for this test. The overall size of the test system is shown in Table VIII.

The overall capabilities of the test system implemented allowed the user to draw arbitary block diagram topologies representing linear time continuous control systems. In the DRAW mode, commands were implemented to allow the user to define and modify block diagrams as well as recentering them within the work area. The section of the PARAMETER mode that was implemented allowed specification of the transfer function polynomials. In the REDUCE mode, only a subset of the EASY5 processing was implemented. This subset allowed the translation of linear time continuous systems only.

User Test.

The primary purpose of the user test was to verify the design decisions made in deriving the specification for the

Table VIII

GRAPHIC Program Test Implementation Size

a) Overlay Sizes:

Overlay	Size: Decimal	Octal
MAIN	15670	36466
DRAW	4210	10162
PARAMETER	3008	5700
REDUCE	9292	22114
SAVE	1301	2425
RESTORE	1196	2254

b) Size During Execution (Including Overhead):

Overlay	Size (octal)
MAIN	36466
DRAW	46655
PARAMETER	44373
REDUCE	60607
SAVE	41120
RESTORE	40747

GRAPHIC system. As such the items that were of greatest interest during the test were whether the user could create the block diagram and modify it using the commands defined and also whether the process provided for doing so seemed logical and convenient to the user. Since the above factors to be evaluated for the test would have subjectively developed answers, the test is best characterized as an experiment rather than a performance measurement.

The subjects used for the test were design engineers from the Flight Dynamics Lab. These subjects would be the prime users of the program and in some cases were individuals who had participated earlier by providing inputs during the problem analysis and requirements definition. The subjects were allowed to use the program with a minimum of instructions. Some instruction was necessary since complete descriptive documentation had not been generated at the time of the test.

During the user test the subjects were first presented with the initial display for the DRAW mode, Figure 13. Using the basic command list in Table IX they were able to experiment with the graphic interface and define systems of block diagrams such as the one in Figure 14. The actual method of producing such a diagram is detailed on a step by step basis in Appendix A. Also it was possible to shift the drawing within the window and cause the miniature to appear automatically when any of the elements fell outside of the work area, Figure 15. Error messages, when required were

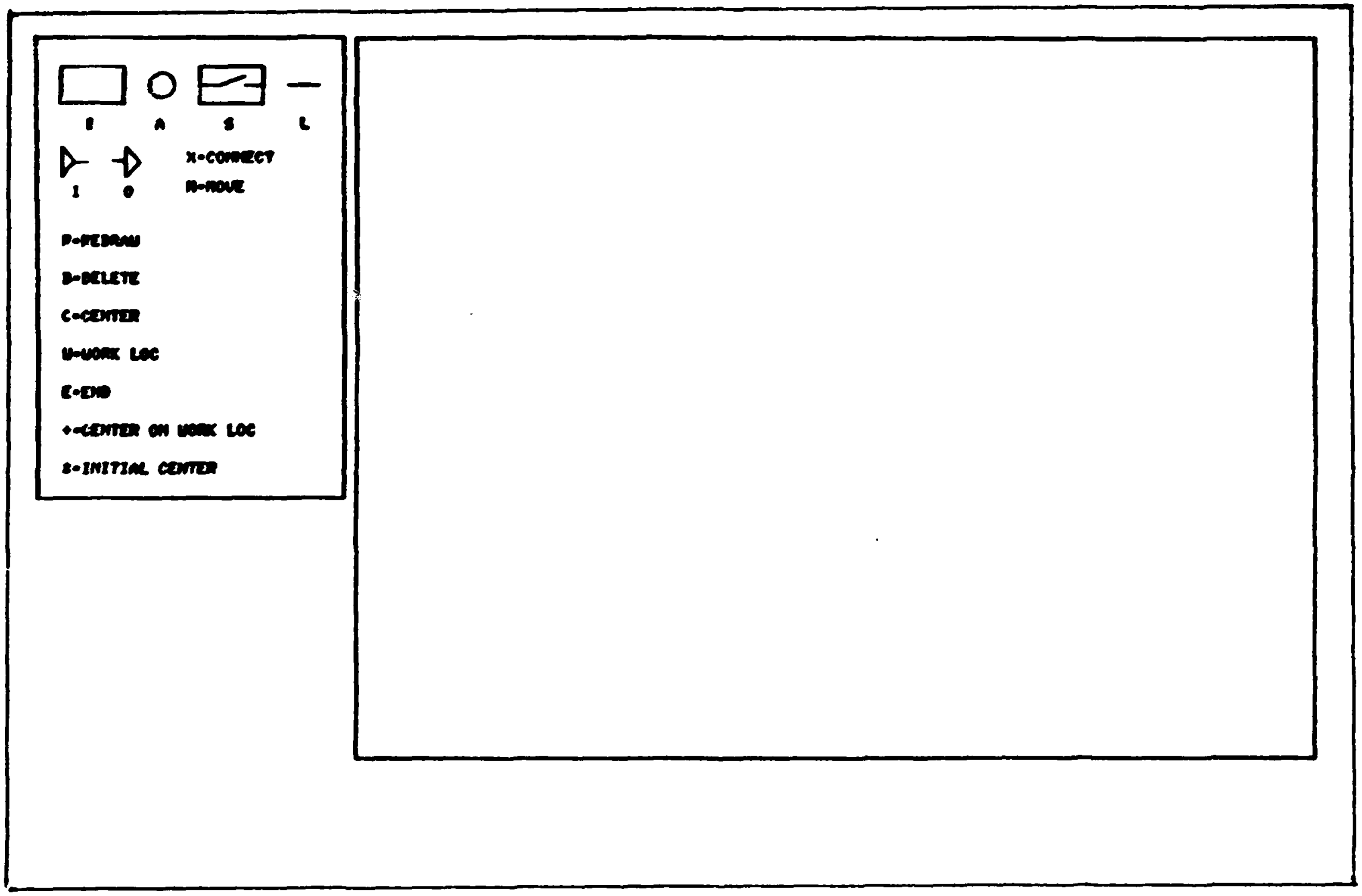


Figure 13. DRAW Mode Initial Display

Table IX

DRAW Mode Commands

Command	Code
Box	B
Sample	S
Adder	A
Input	I
Output	O
Line	L
Move	M
Connect	X
Delete	D
Working Location	W
Center	C
Redraw	R

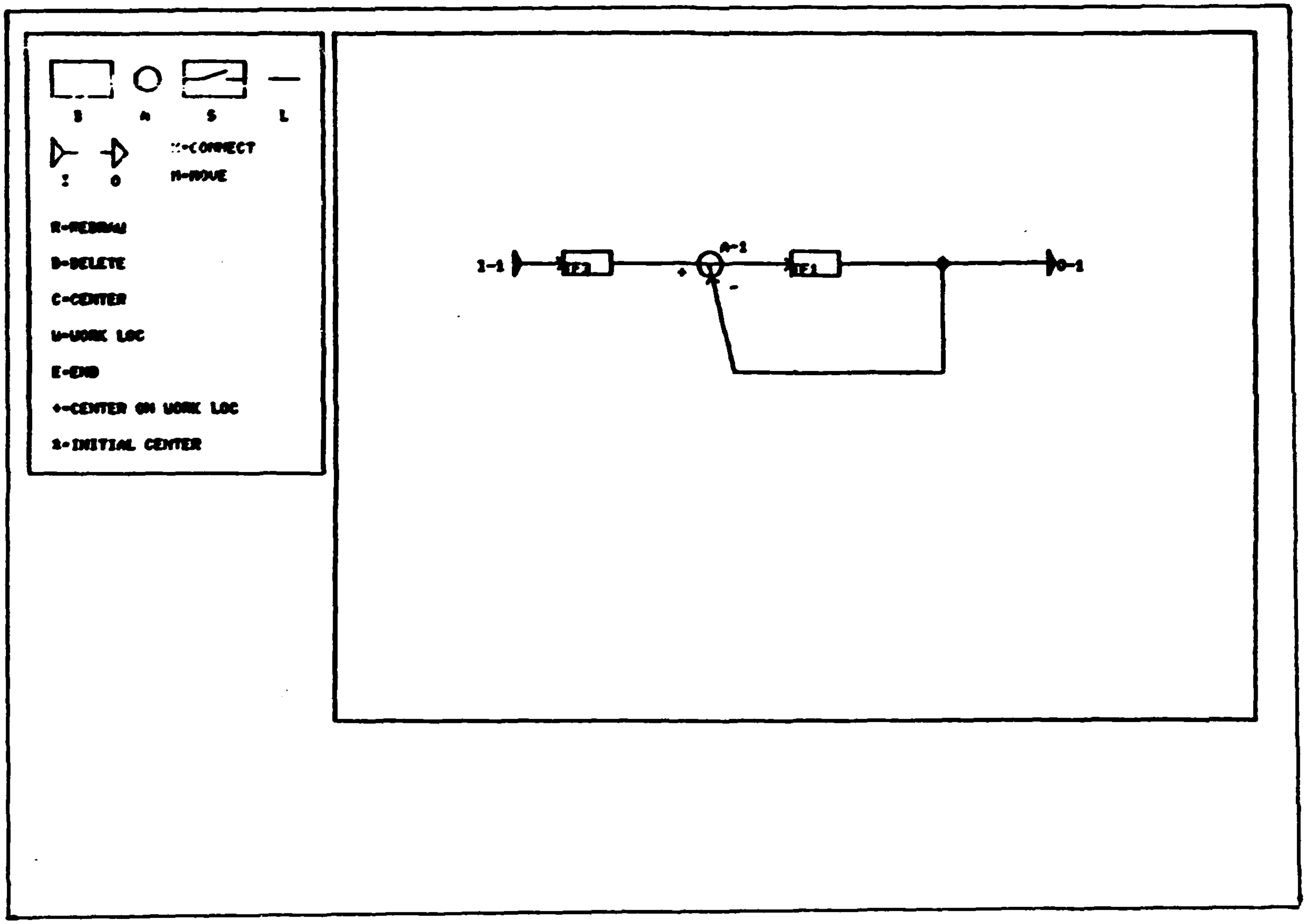


Figure 14. Sample of Block Diagram Definition

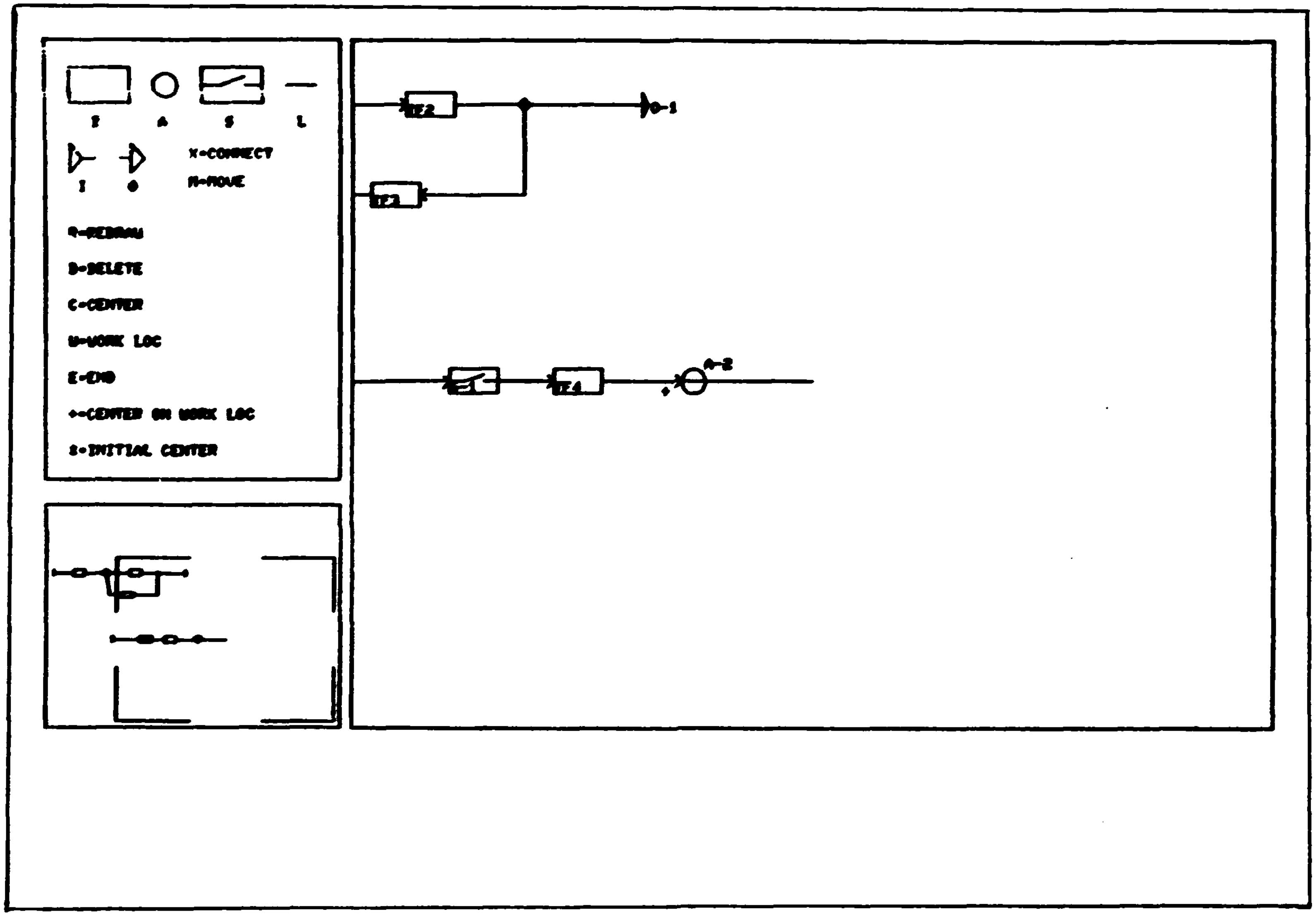


Figure 15. Sample of Miniature Display

scrolled horizontally across each line the text area using a "$" as the start of message delimiter as shown in Figure 16.

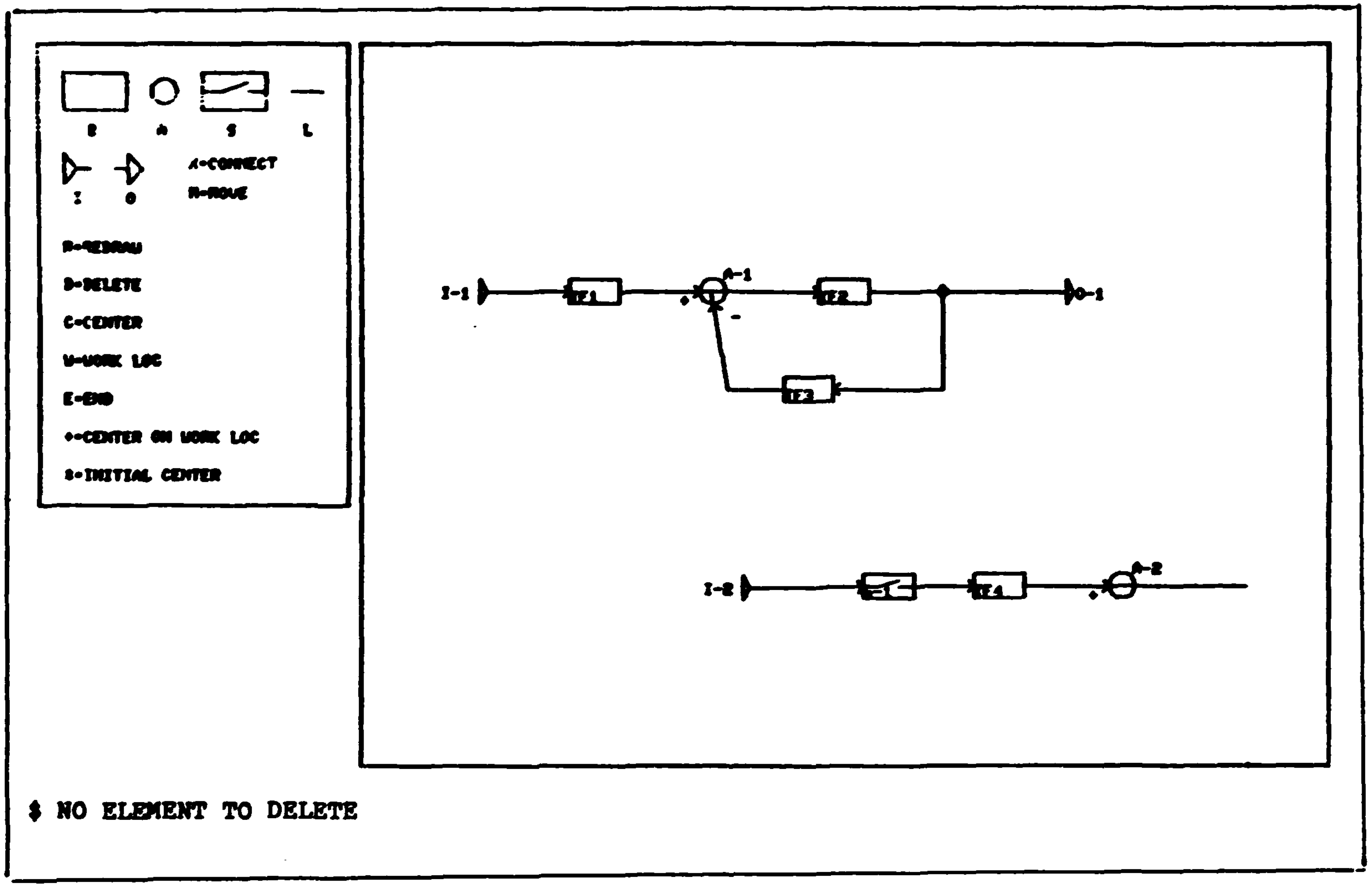
Once the drawing had been defined the user was able to activate the PARAMETER mode. In the test implementation of GRAPHIC all transforms were assumed to be real continuous transforms and all samplers were assumed to have equal sampling periods of one second. The only mode implemented for transform parameter input was where the program automatically polled the user for each transform needed. An example of the format is in Figure 17. Figure 17 corresponds to the input required by the drawing shown in Figure 18.

Once the diagram and its parameters were defined, the user was able to use the REDUCE mode. As mentioned eariler, only a restricted portion of the EASY5 reduction capability was implemented. The system was capable of producing the textual card image definitions for any linear time continuous control system diagram. Figure 19 shows the EASY5 definition produced by the test system from the inputs shown in Figures 17 and 18.

Additionally, the test program allowed the user to operate the SAVE and RESTORE modes. However, for the test implementation only the graphic database was saved.

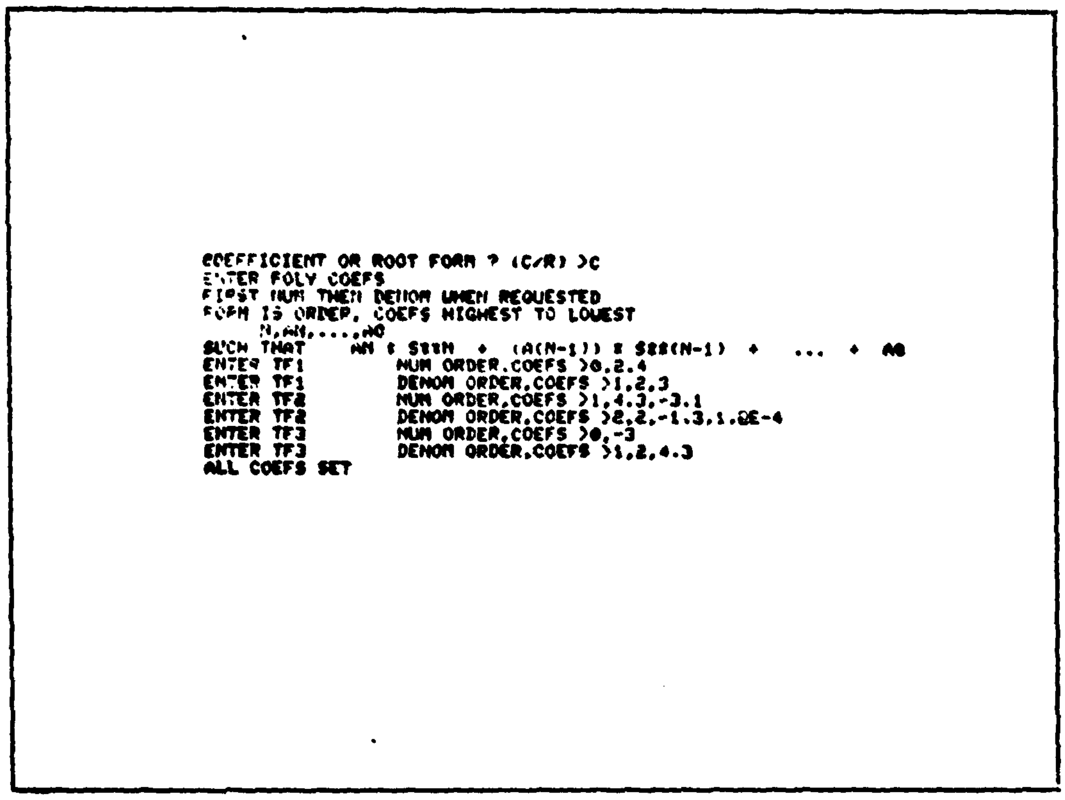
User Test Results

The results of the user test were derived from the activities of the individual users during numerous runs of



```
$ NO ELEMENT TO DELETE
```

Figure 16. Error Message Example



```
COEFFICIENT OR ROOT FORM ? (C/R) >C
ENTER POLY COEFS
FIRST NUM THEN DENOM WHEN REQUESTED
FORM IS ORDER, COEFS HIGHEST TO LOWEST
      N,AN,....,A0
SUCH THAT     AN * S**N  +  (A(N-1)) * S**(N-1)  +   ...  +  A0
ENTER TF1          NUM ORDER,COEFS >0,2.4
ENTER TF1          DENOM ORDER,COEFS >1,2,3
ENTER TF2          NUM ORDER,COEFS >1,4.3,-3.1
ENTER TF2          DENOM ORDER,COEFS >2,2,-1.3,1.0E-4
ENTER TF3          NUM ORDER,COEFS >0,-3
ENTER TF3          DENOM ORDER,COEFS >1,2,4.3
ALL COEFS SET
```

Figure 17. Example of PARAMETER Mode Input

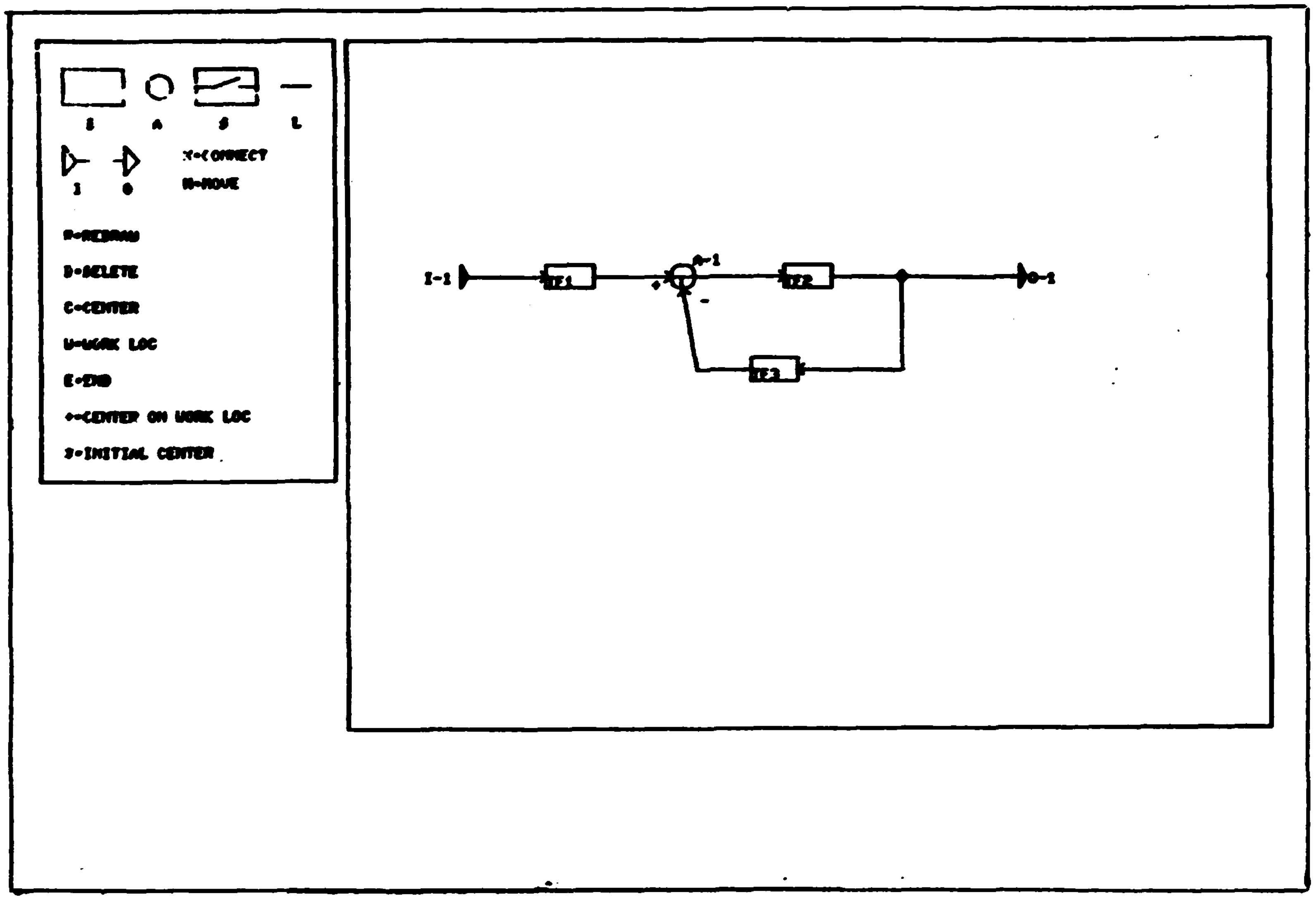
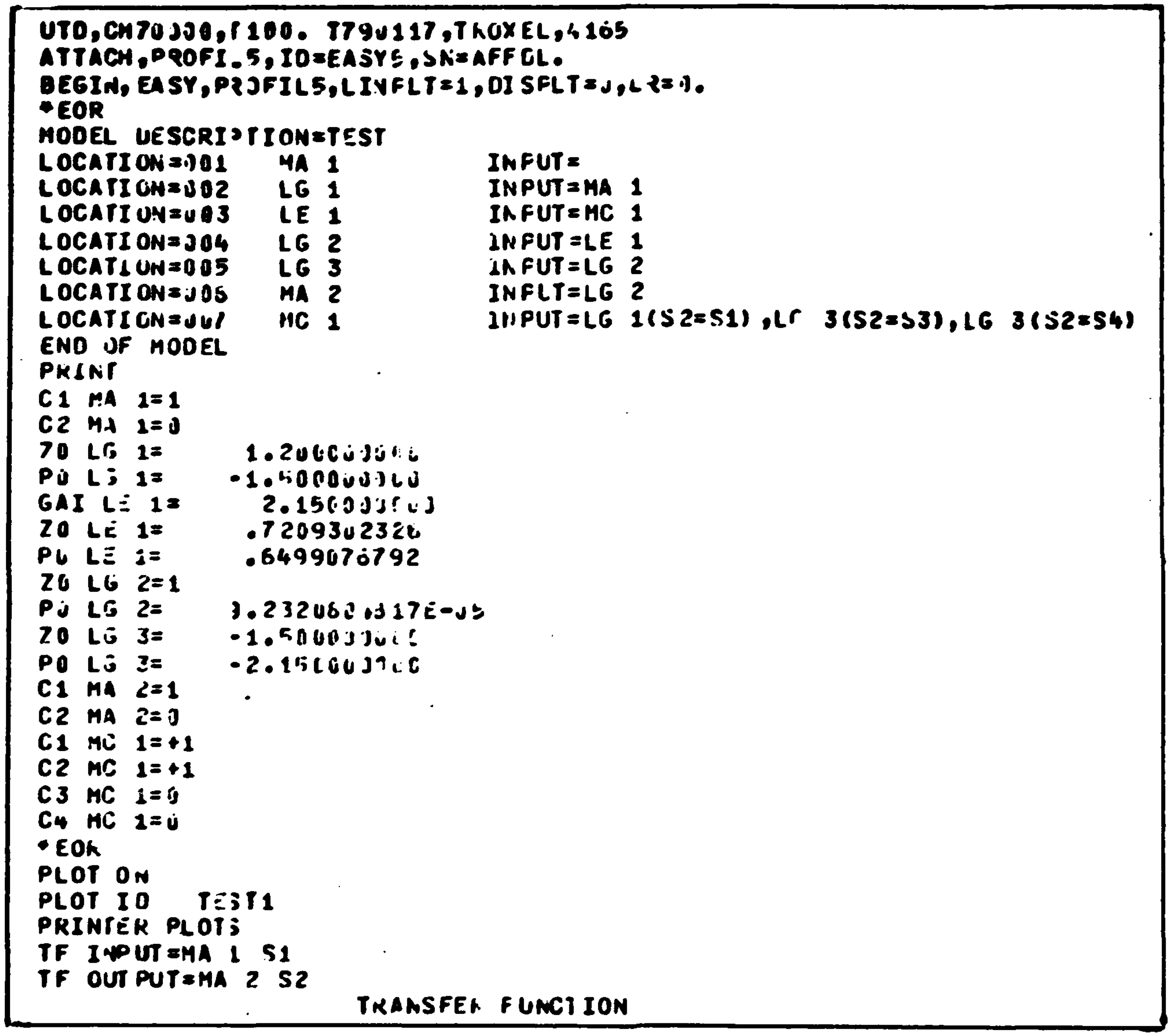


Figure 18. Block Diagram Example



```
UTD,CM70000,T100. T790117,TROXEL,4165
ATTACH,PROFI_5,ID=EASY5,SN=AFFGL.
BEGIN,EASY,PROFIL5,LINFLT=1,DISFLT=0,LR=0.
*EOR
MODEL DESCRIPTION=TEST
LOCATION=001    MA 1          INPUT=
LOCATION=002    LG 1          INPUT=MA 1
LOCATION=003    LE 1          INPUT=MC 1
LOCATION=004    LG 2          INPUT=LE 1
LOCATION=005    LG 3          INPUT=LG 2
LOCATION=006    MA 2          INPUT=LG 2
LOCATION=007    MC 1          INPUT=LG 1(S2=S1),LG 3(S2=S3),LG 3(S2=S4)
END OF MODEL
PRINT
C1 MA 1=1
C2 MA 1=0
Z0 LG 1=     1.2000000000
P0 LG 1=    -1.5000000000
GAI LE 1=     2.1500000000
Z0 LE 1=     .7209302326
P0 LE 1=     .6499076792
Z0 LG 2=1
P0 LG 2=    3.2320603317E-05
Z0 LG 3=    -1.5000000000
P0 LG 3=    -2.1500000000
C1 MA 2=1
C2 MA 2=0
C1 MC 1=+1
C2 MC 1=+1
C3 MC 1=0
C4 MC 1=0
*EOR
PLOT ON
PLOT ID   TEST1
PRINTER PLOTS
TF INPUT=MA 1 S1
TF OUTPUT=MA 2 S2
                  TRANSFER FUNCTION
```

Figure 19. EASY5 Textual Description of Block Diagram

the GRAPHIC test implementation.

During the tests the factors that were of most interest were: 1) questions asked by the test subjects, 2) actions taken by the test subjects that produced results different from what the subject expected, and 3) feelings expressed by the test subjects regarding whether the commands, actions, and graphics seemed logical.

The general results of the user test conducted on the GRAPHIC system are summarized below:

1) The three step process for defining and analyzing a block diagram via the DRAW, PARAMETER, and REDUCE modes was considered a convenient and logical partitioning of the problem.

2) The ability to save and restore the databases was considered essential since it eliminated the need to redefine the block diagram systems upon every reuse of the GRAPHIC program.

3) The division of the terminal face into four distinct areas as previously shown in Figure 3. was considered an acceptable organization for the display.

4) When in the DRAW mode, the complete list of commands in the graphic prompt area was noted as freeing the user from having to memorize the command codes.

5) The use of the first letter of the command mnemonic as the command code was cited as excellent with one exception. The resolution of the conflict between CENTER and CONNECT by assigning the code C to CENTER and the code X to

CONNECT was considered unacceptable. During the user test there was a high incidence of selecting C for the CONNECT operation even with the actual command codes displayed in the prompt area.

6) Correct positioning of the current working location (CWL) prior to using the CONNECT command was not always correctly performed by the users. Since the CONNECT operation affects the elements of the diagram in the vicinity of the CWL, in order to split an output it is necessary to MOVE to the location where the split is to be placed and then perform a CONNECT. It was the MOVE prior to the CONNECT that was often ommitted by the test subjects. The other case of a connection, where an element just drawn is to be connected into the drawing, presented no problem since the act of drawing the element to be connected automatically positions the CWL correctly.

7) The ability to move the drawing around in the work area for defining and viewing different areas of a large diagram was considered essential. The implementation of the CENTER command was considered somewhat inconvenient. It was pointed out by the users that, since the CENTER command required one to point at the position that was to become the center, centering on an element not currently visible required repeated use of the CENTER command. This condition was further aggravated since after each CENTER command there was a delay while redrawing the diagram before the next CENTER command could be issued.

8) The graphic symbols used to depict the various elements in the block diagram were considered acceptable except minor changes in the input and output element symbols were suggested.

9) The automatically generated miniature drawing for large diagrams was cited as being very useful for keeping track of the entire drawing. The use of corner marks to indicate what portion of the miniature was visible in the work area was noted as being useful for rapid correlation between the miniature and work area displays.

Modifications

Based on the results of the user test several modifications were made to the graphic input language. These changes were made to eliminate the confusion that was experienced by the test subjects during the user test.

A review of the user test results shows that items 5, 6, 7, and 8 each identify problems that exist with the initially proposed design for the GRAPHIC system, Appendix A. The discussion that follows addresses each problem identified and the design change that was developed to eliminate that problem.

Item 5 in the test results concerned the confusion between command codes C and X representing CENTER and CONNECT respectively. The initial assumption for the original design was that the use of X as the code for CONNECT could be equated to the use of an X in electronic circuit diagrams to indicate a connection between two wires that crossed. This assumption proved false as was demonstrated in the test. Since it was noted during the test that CONNECT commands were used more often than CENTER commands, the letter C was assigned as the code for CONNECT. The command code for CENTER was also modified. Changes to the CENTER command are explained in conjunction with item 7 below.

Test result item 6 addresses another problem associated with CONNECT. This problem arises when the user forgets to first position the CWL with a MOVE command before attempting to split outputs via the CONNECT command. This is a clear example of the process model and the user model not matching. The process model, as controlled by the input language, represents this action as a two step operation. On the other hand, the user views it as a one step operation, simply connect. Further analysis indicated that the first step, the MOVE step, in the computer process model is associated solely with positioning the CWL. Since the concept of a CWL is not consciously present in the user's mental model of the actions required for drawing a diagram, clearly it is the MOVE step which must be eliminated. The approach taken was to assume a MOVE command was issued and handle it automatically, if the crosshair location is not at the CWL when the CONNECT command was received. This approach solves the problem by causing the process model and user model to coincide.

Item 7 of the test results identified a problem with the CENTER command. The manifestation of the problem was that the user could not center an object in the work area with a single command if the element was not already visible in the work area. The solution to this was to expand the power of the CENTER command. Specifically, when pointing to the element that was to be centered in the work area, the user would now be allowed to select any element that appeared in either the work or miniature areas. This would allow direct one command positioning to any element in the diagram. Additionally, two other specialized options of the CENTER command were created. One allowed centering on the initial default window, while the other allowed centering on the CWL.

The symbol representation problem identified in item 8 was just a graphical depiction problem. It is a simple example of the problems that occur when the programmer is allowed to specify the aspects of the interface instead of the user. The solution to this was simply to redefine the graphic symbols as the user expected them to appear.

The change in the graphic symbols was effected as soon as the problem was discovered. This was done to prevent the graphic symbology problem from overshadowing any of the other aspects of the test. The old symbols and the new final versions are shown in Figure 20. Because the change occurred at a very early stage, all figures in this report show only the final graphic symbols for these elements. The other changes mentioned above were accomplished as modifications to the proposed design after the test was completed. The complete details of these changes are documented in Appendix B.

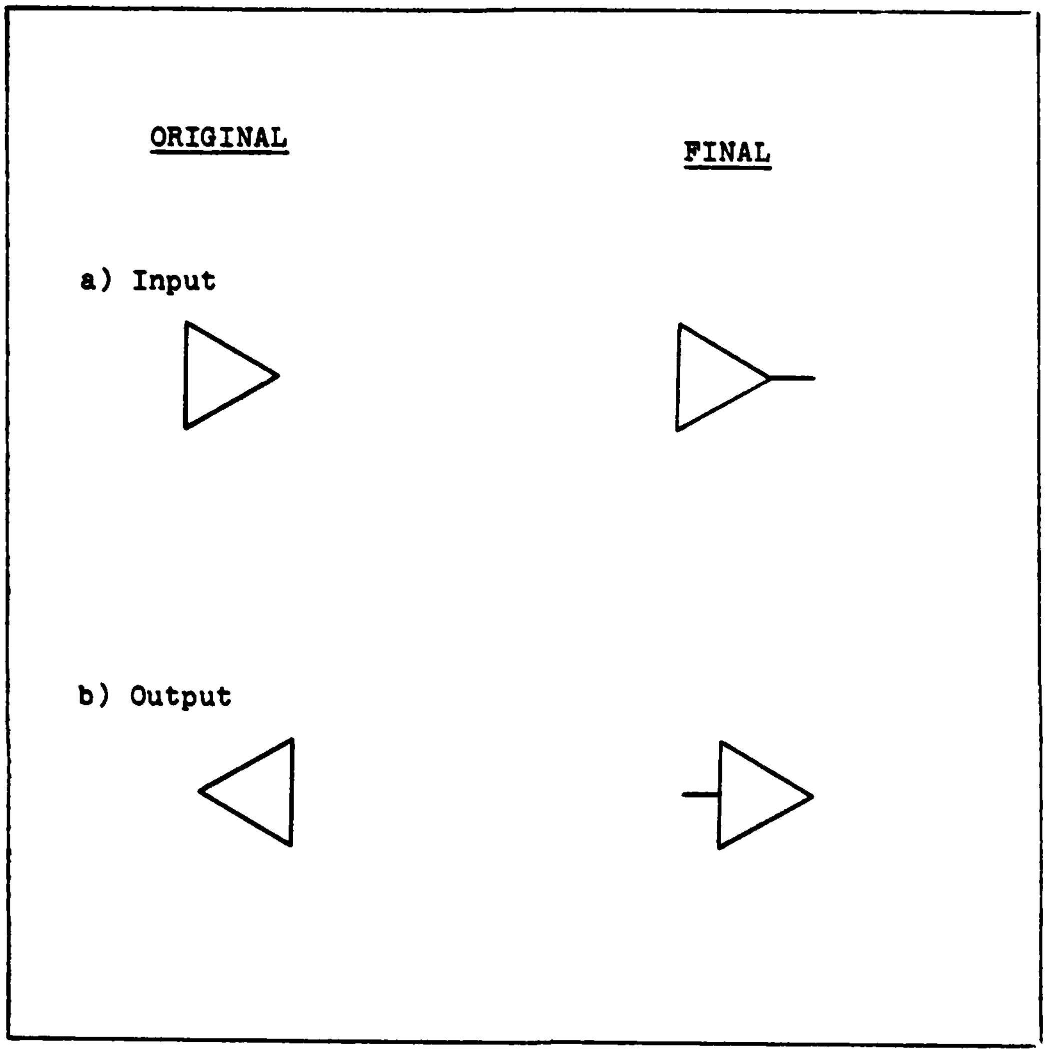


Figure 20. Representation of Input and Output Symbols

VI. Conclusions and Recommendations

The GRAPHIC system as designed represents an effective computer aided design tool. It allows the rapid specification of arbitrary topology contol systems in the form of block diagrams and then translates these diagrams into forms acceptable to other programs such as EASY5 and TOTAL for further analysis.

The implementation of the graphic input language for this system is well suited for the process of creating the block diagrams. The interface as designed matches the level of concentration required by the user for CRT definition of a block diagram with that required for a paper sketch definition of the block diagram. In this way the interface retains a natural feeling for the user instead of requiring conscious thought for the selection of the correct input command. The use of graphics successfully hides unneeded information, such as orientation angles of symbols, while displaying the significant items, the signal flow interconnections, in their natural diagrammatic form. This makes optimum use of the human pattern recognition capabilities available for error detection since any errors are completely depicted in the visible drawing on the CRT.

The user test sucessfully demonstrated that an effective graphical method can be developed for use in specifying control system block diagrams for computer analysis. It also demonstrated that this graphical method can be designed to allow rapid block diagram definition while retaining enough aspects of the normal non-computer sketching method to continue to feel natural to the user.

Due to the positive results obtained during the user test it is recommended that a fully functional version of the GRAPHIC program be implemented. This system should be created from the design specified by Appendices A and B. The test implementation as documented in Appendix D should serve as the foundation for this implementation.

Although the GRAPHIC system greatly improves the ease with which a user can generate block diagrams for analysis, there are still many areas where further investigation would be fruitful. These areas are summarized below:

1) The GRAPHIC program currently is designed only to interface with the Tektronix 4000 series terminals. Any hardcopy of the diagrams analyzed is produced through these terminals. It would be desirable to extend the design to allow for other interactive graphic terminals and also to allow the diagrams to be copied to noninteractive devices such as plotters.

2) As was mentioned earlier, the GPAPHIC support library CORE allows drawing in three dimensions. It would be desirable to investigate the use of the third dimension for representing varying levels of detail. Conceptually this would allow one to step back and view a system under analysis as several high order black boxes or zoom in for a [illegible] detailed representation of the individual elements

within the black box.

3) The entire GRAPHIC system should be integrated with the TOTAL system (Ref 12) to allow complete definition and analysis of control systems as though the two packages were one consolidated computer aided design tool. Specifically, the consolidation should attempt to eliminate any conscious user thought involved with transferring data between the GRAPHIC and TOTAL packages.

Bibliography

1. Air Force Institute of Technology. Digital Computer Manual for Faculty and Students of the School of Engineering (4th Edition). Wright-Patterson AFB, Ohio: AFIT, August 1978.

2. ASD Computer Center. Battelle Disk File Manipulation Routines User's Guide (Revision B). Wright-Patterson AFB, Ohio: ASD Computer Center, July 1978.

3. ASD Computer Center. CDC NOS/BE User's Guide (Revision F). Wright- Patterson AFB, Ohio: ASD Computer Center, August 1979.

4. ASD Computer Center. Cyber Control Language. Wright-Patterson AFB, Ohio: ASD Computer Center, December 1977.

5. ASD Computer Center. INTERCOM Guide (Revision A). Wright-Patterson, Ohio: ASD Computer Center, September 1976.

6. ASD Computer Center. Subprogram Library Guide (Revision D). Wright-Patterson AFB, Ohio: ASD Computer Center, January 1978.

7. Boeing Aerospace Company. EASY5 User's Manual for Control System Simulation. Document No. D 180-19147-3, Contract F33615-79-C-3407. Seattle, Washington, Boeing Areospace Company, 1979.

8. Control Data Corporation. Fortran Extended Version 4 Reference Manual. Pub. No. 60497800, Revision D. Sunnyvale, California: Publications and Graphics Division, 1978.

9. Control Data Corporation. INTERCOM Version 4 Reference Manual. Pub. No. 60494600, Revision E. St. Paul, Minnesota: Publications and Graphics Division, 1978.

10. DeMarco, Tom. Structured Analysis and System Specification. New York: Yourdon Inc.,1978.

11. Foley, James D. and Victor L. Wallace. "The Art of Natural Graphic Man-Machine Conversation," Proceedings of the IEEE, 62: 462-471 (April 1974).

12. Larimer, 2nd Lt. Stanley J. An Interactive Computer-Aided Design Program for Digital and Continuous Control System Analysis and Synthesis. MS thesis. Wright-Patterson AFB, Ohio: Air Force Institute of Technology, March 1978. (AD A055 418).

13. Larimer, Stanley J. and Gary B. Lamont. "An Interactive CAD Program for Control System Analysis and Synthesis," Proceedings Sixteenth Annual Allerton Conference on Communication, Control, and Computing, 287-295 (October 1978).

14. Miller, George A. "The Magic Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information," Psychology Review, 63 (2): 81-97 (March 1956).

15. Newman, William M. and Robert F Spoull. Principles of Interactive Computer Graphics (Second Edition). New York: McGraw-Hill Book Company, 1979.

16. Spence, Robert and Mark Apperley. "The Interactive-Graphic Man-Computer Dialogue in Computer-Aided Circuit Design," IEEE Transactions on Circuits and Systems, CAS-24: 49-61 (February 1977)

17. "Status Report of the Graphics Standards Planning Committee of ACM/SIGGRAPH," Computer Graphics, 11 (3) (Fall 1977).

18. Tektronix, Inc. Plot 10 Terminal Control System User's Manual. Document No. 062-1474-00. Beaverton, Oregon, Information Display Division, Tektronix, Inc., 1974.

19. Troxel, Donald E., et al. "CORE," unpublished report for course MA 6.86, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1979.

20. Voltz, Richard A., et al. "COINGRAD - Control Oriented Interactive Graphical Analysis and Design," IEEE Transactions on Education, E-17: 143-152 (August 1974)

21. Young, Kenneth Royce. The Design of Automatic Control Systems Using Interactive Computer Graphics. Ph.D. Dissertation, The University of Texas at Austin, 1974. Ann Arbor, Michigan, University Microfilms International, 1979.

22. Yourdon, Edward and Larry L. Constantine. Structured Design (Second Edition). New York: Yourdon Press, 1978.

APPENDIX A

USER'S MANUAL

FOR

GRAPHIC

Preface

This appendix documents the complete design specification for the GRAPHIC system. It is structured as a User's Manual for two reasons. The first reason is to highlight the features that will be available as part of the GRAPHIC package. The second reason is to enable actual examples of user inputs and outputs to be specified. This approach of specifying the system design via a User's Manual was considered to provide the best means of allowing the system to be visualized prior to its implementation.

Contents

Contents

List of Figures

List of Tables

USER'S MANUAL FOR GRAPHIC

I. General Description

GRAPHIC is designed to allow the user to rapidly define complicated control systems for further analysis. It does this by allowing the user to draw the desired control system in graphical block diagram form on the face of a CRT. In this way the GRAPHIC system avoids the need for complicated textual descriptions of large control systems while capitalizing on the human's inherent pattern recognition capability for drawing and correcting the graphical description. Once defined in graphical form, the GRAPHIC program can be used to find the equivalent transfer function between any two points in the diagram. Also, if desired, the diagram can be translated into its equivalent textual description for analysis by the EASY5 program (Ref #7).

Overview

The GRAPHIC system is designed to operate in a timesharing mode with the user at a Tektronix 4000 series terminal equiped with a graphic crosshair. The terminal display area is divided up into four areas as shown in Figure 1. Each area is used only for the purpose indicated.

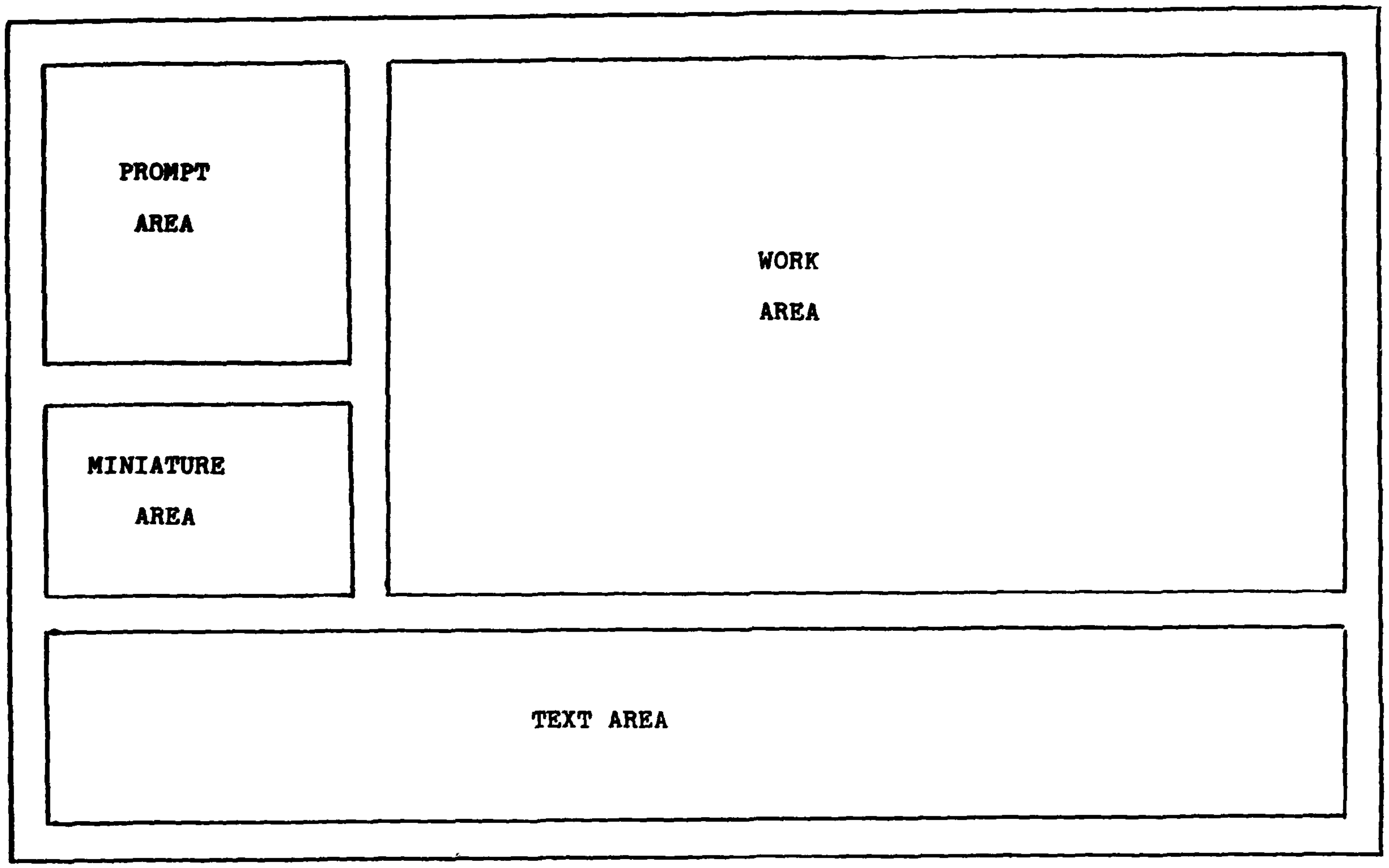


Figure 1. Terminal Display Organization

There are six major modes of operation in the GRAPHIC system. These modes are described in detail in chapter two and are summarized below:

The DRAW mode allows block diagram topologies to be created and modified.

The GROUP mode allows groups of elements to be created so that the entire group can be referenced by one command.

The PARAMETER mode allows the mathematical parameters of the individual transfer functions to be specified. The sampling periods are also specified in this mode.

The REDUCE mode allows the block diagram to be reduced to its equivalent transfer function or reduced to its equivalent textual description for further analysis by the EASY5 program.

The SAVE mode allows the block diagram and any mathematical parameters describing it to be stored as a permanent file.

The RESTORE mode allows a permanent file created in the SAVE mode to be used for input to restore the previous definition of a block diagram and its numeric parameters for further modification and analysis.

General Operation

The normal method of using the GRAPHIC program for block diagram analysis would be to select the different modes of operation in the order described below.

Upon initiation the GROUP mode would usually be selected first. In this mode any groups of the symbols the user desired to create could be defined. This mode is optional and no groups of symbols need to be defined. The next mode would be the DRAW mode. This mode would allow the user to define the block diagram he desired. Next, the PARAMETER mode would be selected to specify the switch sampling periods and the mathematical transfer functions for the elements of the diagram. After these modes had been completed the REDUCE mode would be selected to accomplish the form of reduction desired by the user. Finally, the SAVE mode would be selected if the user desired to reuse the current block diagram at a later date.

If the drawing had been stored perviously, the RESTORE mode could have been selected before the GROUP mode. This method allows the reuse of a diagram from a previous activation of the GRAPHIC program.

AD-A080 362 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
GRAPHICAL INPUT METHODOLOGY FOR COMPUTER AIDED ANALYSIS OF CONT--ETC(U)
DEC 79 D E TROXEL
UNCLASSIFIED AFIT/GE/MA/79D-1 NL

2 OF 2
AD
A080362

END
DATE
FILMED
3-80
DDC

Notation

Throughout this manual numerous examples of input and output appear. For presentation of these examples the following convention was used to distinguish between computer generated output and user generated input. The computer generated output is always shown in capitial letters and the user generated input is always shown in lower case letters.

Example:

```
COMPUTER OUTPUT > user input
```

Additionally, due to the syntax selected for the user interface, most of the lines of text change from computer output to user input at the delimiter ">".

When further explanatory notes are needed to help clarify the example they are shown in parenthesis.

Example:

```
COMMAND > draw          (Select draw mode for diagram
                         definition.)
```

II. Modes of Operation

As mentioned in chapter one there are six modes of operation in the GRAPHIC program. These modes are DRAW, GROUP, PARAMETER, REDUCE, SAVE, and RESTORE. The operation of each mode is described in detail in this chapter.

Draw Mode

In order to define the system to be analyzed the user must draw a picture of the system's block diagram on the CRT.

To form the desired drawing the user must use the graphic crosshair to indicate where the symbol is to go and the keyboard to indicate what symbol is to be drawn. Every symbol is positioned in the same manner. First move the graphic crosshair by using the thumbwheels and then when the desired position is reached enter the one letter code for the desired symbol. The commands allowed and their letter codes are shown in Table I.

This method of producing the block diagram a symbol at a time has been designed to require only one input, crosshair position, per symbol. Because of this all symbols are produced at a preset and fixed size.

The elements of the block diagram must be drawn in the direction of signal flow. When it is necessary to continue from another point, the move command should be used. While drawing is under way the system keeps track of the current working location (CWL) and updates it after each symbol is

Table I

DRAW Mode Commands

Command	Code
Box	B
Sampler	S
Adder	A
Input	I
Output	O
Line	L
Move	M
Connect	X
Delete	D
Working Location	W
Center	C
Redraw	R

drawn. This working location is used as the starting point when a symbol is requested and the crosshair is used as the end point if an end point is needed.

The box, adder, sampler, input, and output symbols must be connected to a line and not directly to each other. Drawing of symbols must always proceed in the direction of signal flow. Arrowheads to help clarify the direction of signal flow will be generated periodically and automatically by the system when the topology of the diagram becomes unambiguous. These arrowheads will appear at the inputs to box, adder, and sampler symbols.

Box Command - Code:B. The crosshair location is not needed for drawing the box, only the letter "B" is used. The box is drawn fron the current working location, in the direction that the last line was drawn. A new working location is established on the far side of the box when it is complete. Figure 2 contains an example how the box and all other symbols are oriented and created.

The box symbol is used for all transfer functions. As each box is drawn the user will be queried for the type of box being defined. The square text cursor will appear in the lower left corner of the box. The type name is then entered and following the type entry the box will be numbered automatically.

The general type, TF, can be specified for arbitrary continuous or discrete transfer functions. Specialized types for use with EASY5 reduction can also be specified. These

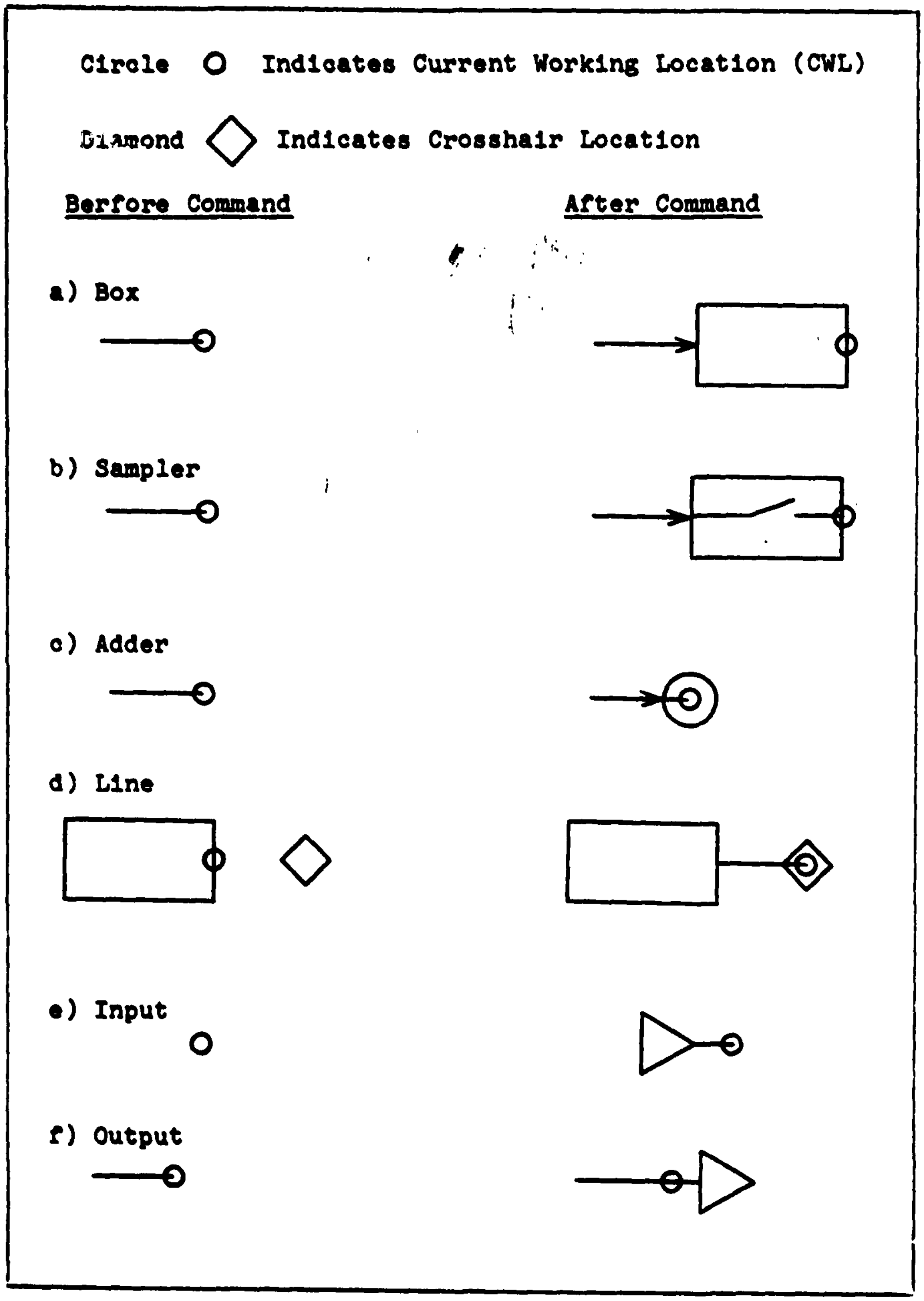

Circle Indicates Current Working Location (CWL)
Diamond Indicates Crosshair Location
Berfore Command
After Command
a) Box
b) Sampler
c) Adder
d) Line
e) Input
f) Output

Figure 2. Example of Individual Symbol Creation

special types include FORT, AV, LO, and SD (Ref #7).

Parameters defining the specifics of each box are input in the parameter mode.

Adder Command - Code:A. The crosshair location is not needed for drawing the adder, only the letter "A" is used. The adder is drawn centered on the current working location and this location is not changed. See Figure 2 for an example of adder creation.

This symbol is used whenever two or three inputs must be added together. If more than three inputs are to be added at any one place, two or more adders must be used in series.

The sign of the input is requested as each input to the adder is defined. The square text cursor will appear next to the input and the user must input either a "+" or "-".

Adders are numbered with An numbers for identification starting with A1.

Sampler Command - Code:S. The crosshair location is not used for drawing the sampler, only the letter "S" is used. The sampler will be drawn from the current working location in the same direction as the previous line. The working location will be updated to the far side of the sampler symbol. See Figure 2 for an example of sampler creation.

This symbol is used whenever digital sampling is required.

As each sampler is created it will be numbered for identification with a Sn number, starting with S1.

Input Command - Code:I. The crosshair location and the letter "I" are used to draw the input symbol. The input symbol is upright and to the left of the crosshair location. The current working location is not used and a new current working location is established at the crosshair location. See Figure 2 for an example of input symbol creation.

This symbol is used to indicate all input nodes.

As each input symbol is produced it will be numbered for identification with a In number, starting with I1.

Output Command - Code:O. The crosshair is not used for drawing the output symbol, only the letter "O" is needed. The symbol is drawn upright and to the right of the current working location. See Figure 2 for an example of output symbol creation.

This symbol is used to indicate all output nodes.

As each output symbol is produced it will be numbered for identification with an On number, starting with O1.

Line Command - Code:L. The crosshair location and letter "L" are used in drawing the line. The line is drawn from the current working location to the crosshair location and the current working location is changed to the crosshair location. See Figure 2 for an example of line creation.

The direction that the line is drawn in, from current working location to crosshair location, always is taken as the direction of signal flow through the block diagram.

Move Command - Code:M. The crosshair location and the letter "M" are used for this command. The current working location is changed to the crosshair location.

The move command is used to change the current working location so that the definition of the block diagram can continue from some arbitrary point that is not connected to the last symbol that was drawn.

Connect Command - Code:X. The crosshair location is not used, only the letter "X" is needed.

The connect command is used to specify that at the current working location a connection is to be formed with a previously drawn element. If this is a connection to split an output path then a connection diamond will be drawn. For other types of connections no diamond will be drawn. Connections without a diamond symbol will most often be used to continue a drawing along a path that was interrupted earlier or for the termination of a path in an adder symbol. If the connection is to an adder symbol the appropriate + or - sign will be requested as explained under adder commands.

In all cases if a successful connection is formed the current working location will be at the crosshair location. If an error occurs, such as no symbol to connect to, the terminal will beep and the current working location will not be changed.

End Command - Code:E. The crosshair location is not used, only the letter "E" is needed. This command is used to terminate the DRAW mode of operation and return to the primary command level where another mode may be selected.

Delete Command - Code:D. The crosshair location and the letter "D" are used in this command. The element pointed to by the crosshair will be logically deleted from the database and the drawing will be marked with two X's to indicate the element is no longer an active part of the diagram. On subsequent redraws of the diagram the element deleted will not be shown at all. The current working location is unchanged.

If, under the crosshair location, no element of the drawing can be found or if more than one element is found this is considered an error and the terminal will beep indicating that the command was ignored. If a deletion is still desired the crosshair location should be changed slightly to better designate the desired element and then the command reentered.

Redraw Command - Code:R. The crosshair location is not used and only the letter "R" is needed. A redraw causes the terminal screen to be erased and all current elements in the database for the block diagram are drawn on the CRT. The current working location is unchanged. The information displayed in the text area is not redisplayed.

This command is used to remove from view all items that have been marked by X's from previous delete commands.

Working Location Command - Code:W. The crosshair location is not used and only the letter "W" is needed. When the command is received the square text cursor is displayed with the lower left corner of the cursor indicating the position of the current working location.

To bring the crosshair back for further drawing of block diagram symbols enter a carrage return. This will cause the crosshair to be redisplayed.

Center Command - Code:C. The crosshair location is used to indicate the spot in the existing diagram that is to be centered in the work area. The current working location is relocated along with the entire diagram so that it remains on the last symbol in the same relative position to the elements of the diagram as before the command.

When the block diagram becomes larger than the work area the center command is used to shift the existing drawing within the work area to create more room for defining elements. If any of the existing diagram gets shifted out of the work area, a miniature of the entire diagram will be produced in the miniature area. The miniature is produced automatically unless inhibited by the value of the flag MINIATURE.

The miniature if produced is a copy of the diagram at the time of centering and is not updated by further drawing in the work area. The miniature is updated only when another center command or a redraw command is issued.

Group Command - Code:n. Both the crosshair location and the numeric code n are used to draw the user group of elements. The code can range from 0 to 9. It indicates which of the groups created by the GROUP mode commands is to be drawn. The CWL is moved to the point on the group defined as the final CWL location when the group was predefined.

User groups are a method of conveniently drawing a group of symbols that are going to be repeated several times within a diagram. Once defined as a group, the symbols can be copied into the drawing at any location by giving the group number as the command code. This works exactly the same as when a code letter is given except more than one symbol is drawn.

Elements within the user group exist as individual items and can be modified as though they had been drawn individually.

Group Mode

The GROUP mode is used to define collections of elements so that the entire group can be referenced by one command. These groups are designed to be collections of elements, such as a subsystem, that would appear repeatedly in a block diagram that is to be defined. By using the GROUP mode and defining a user group, the entire collection of elements can be referenced in the DRAW mode with one command instead of having to position each element individually.

To define a user group of elements, the collection of elements is drawn in the work area in the same manner as entire diagrams are created in the DRAW mode. When all elements of the group have been drawn an E is entered to indicate that the graphical definition of the user group is complete. Next, any numeric parameters for box or sampler elements in the group can be given default values. This is done using the same methods employed in the PARAMETER mode. When all of the desired defaults have been entered, the user group is fully defined. It is now stored in the database and is also available for use in the DRAW mode when referenced by its group number.

The individual graphical elements and any numeric defaults in the user group can be altered in the DRAW and PARAMETER modes as described in their respective sections. Any alteration during the DRAW or PARAMETER modes effects only one particular occurrence of the group and not all occurrences of the group. To actually redefine the diagram or defaults for a group the GROUP mode must be used.

Example of group definition:

```
COMMAND > group
USER GROUPS NUMBER 1,3,7 EXIST
ENTER GROUP NUMBER TO BE DEFINED OR REDEFINED > 2
DRAW DIAGRAM OF GROUP 2 ELEMENTS NOW
```

enter: (The user must draw the elements he wants to become the new group. The point where the drawing starts is the point that will be connected to the current working location (CWL) whenever the user group is involked by typing its number and the CWL that exists when the definition of the user group is completed is the relative location of the CWL after the user group is involked. When the END command is entered to leave the drawing mode the user group is complete and its topology is established.)

note: (All drawing commands except the user group command are available for defining the group.)

```
ANY DEFAULT NUMERIC PARAMETERS DESIRED ? (Y/N) > y
COEFFICIENT OR ROOT FORM ? (C/R) > c

note: (Inputs are in the same form and sequence as in
      the PARAMETER mode. Defaults can not be
      assigned to any of the specialized EASY5
      elements. Elements do not need to have
      defaults established for them since they can
      be specified later in the PARAMETER mode.)

ALL PARAMETERS DEFINED.
USER GROUP NUMBER 3 ESTABLISHED
COMMAND >
```

A maximum of ten user groups can be defined and stored for use at any one time. These groups would each have a one digit reference number ranging from Ø to 9. If more groups than ten are required, they must be used in succession. This is done by defining the first ten, then creating some of the block diagram in the DRAW mode with these groups. When the additional groups are needed, the original groups that are no longer needed for further drawing are redefined. This operation effects only the list of groups available for use in the draw mode. It has no effect on groups that have been referenced and already appear in the main diagram. Once a group is added to the main diagram in the DRAW mode, the elements within that group exist as individual elements and all association with the group definition is lost.

Parameter Mode

The PARAMETER mode is used to define the numeric values for transfer functions and sampling rates. In this mode all rates, functions and specialized parameters are defined to complete the mathematical specification of the diagram created in the DRAW mode.

A standard naming convention is used to allow any parameter to be referenced directly. This allows specification of values in any order that is convenient. The naming convention uses the labels from the graphical drawing and appends individual letters as needed to properly identify the parameters to be entered. Samplers are designated using exactly the same labels as those that appear in the drawing. Transfer functions for individual box elements are identified in three methods. The three methods each reference different parts of the function. The label as it appears in the drawing is used to refer to the entire function. The label with an N appended to it is used to refer to just the numerator portion of the transfer function. Similarly, the label with a D appended to it refers to the denominator portion. A list of the naming conventions appears at the end of this section. Parameters for the specialzed EASY5 elements are identified using the normal EASY5 terminology. This terminolgy is explained in the User's Manual for EASY5 (Ref 7).

These parameter names are used to reference items individually and in random order. This is done when less than the complete list of required parameters is being established during this use of the PARAMETER mode. Additionally, the names are used when two sets of parameters are to be set equal to each other. The method for doing this is explained later in this section.

At any time during the parameter mode the calculator feature is available for use in scratch pad operations. This feature functions exactly as the TOTAL calculator and is activated by typing "C" in place of any single input. It allows scratch pad calculations without affecting the input in progress. A full description of calculator operations appears in the TOTAL User's Manual (Ref #12).

The PARAMETER mode expects all polynominals for transfer functions to be input in either coefficient or root form. The desired form is established after the PARAMETER mode is activated. Individual parameters can be specified using the standard names or the system can be requested to automatically sequence through the needed parameters. If a parameter is to be redefined, the complete list of values does not have to be reentered. Instead, an * can be substituted for any numeric value that is to remain unchanged. Additionally, values can be set equal to previously defined values by equating their standard names.

Examples for the generalized transfer functions, those box elements with TFn numbers, are shown below:

Example:

```
COMMAND > parameter
AUTOMATIC SEQUENCING ? (Y/N) > n
COEFFICIENT OR ROOT FORM ? (C/R) > r
ENTER
LABEL, ORDER, ROOT REAL PART, ROOT IMAGINARY PART >
tfln,*,*,*,3,*    (Redefine real part of 2nd root)
```

Example:

```
   COMMAND > parameter
   AUTOMATIC SEQUENCING ? (Y/N) > y
   COEFFICIENT OR ROOT FORM ? (C/R) > c
   ENTER ORDER, COEFFICIENTS FOR
   TF1N > 1,2.8,1
   TF1D > 2,1.2,4.3,-6,5
   TF2N > tf1              (Sets denominator also)
   TF3N > tf1n
   TF3D > 1,5,7
   ALL VALUES SET
```

Parameters for EASY5 specialized boxes will be requested via individual prompts. Proper utilization of these complex multi-input and multi-output elements will require a detailed knowledge of and familiarity with EASY5 (Ref #7).

The user will be prompted for each EASY5 required value in accordance with the particular element types used. Prompting will be in EASY5 termonology.

Example:

```
   ENTER VALUES FOR ELEMENT AC 1
   ALPHA=1.45
   BETA=2e-3
```

When special additional information is needed for AV,SD,LO,LD, and FORT type elements of EASY5 then the information will be requested at the terminal in EASY5 card image form.

Sampler elements require that the user specify the sample period. These periods will be input in seconds or fractions of seconds.

Example:

```
ENTER SAMPLING PERIODS:
S1=1e-3
S2=.01
```

Rates can be set equal to each other by supplying the name of a sampler for which the period has allready been defined.

Example:

```
S3=s1
```

Rates that are not integer multiples of each other throughout the system will be changed to the nearest multiple so that they will all be integer multiples of an arbitary basic rate.

Only single rate systems can be reduced on line in the interactive mode as described in the section on REDUCE mode operation.

Reduction Mode

Reduction mode is used to reduce the current block diagram to find the equivalent transfer function between a specified input and output.

First the diagram topology is checked to insure that no unconnected ends exist which are not either designated as inputs or outputs. Then the database is checked to insure that all transfer function and sampling period values have been specified. If errors are detected a list of errors is output to the terminal and the reduction mode is terminated.

If no errors are found the user is requested to supply the input and output points he wants the transfer function calculated between.

Example:

```
REDUCTION BETWEEN INPUT(.) AND OUTPUT(.)
INPUT NODE IS > 1
OUTPUT NODE IS > 2

result: (The equivallent transfer function will be
        displayed on the terminal in both coefficient
        and factored form. The value will also be
        stored in the database as transfer function
        answer n (TAn).)
```

There is space to store up to 20 answers from reductions of diagrams. These functions, TAn, can be used in later drawings as equations for individual elements by giving the name TAn when requested for the degree of the numerator in the parameter mode.

To indicate an input node point that is within the drawing instead of one of the input symbols give its complete alphanumeric description instead of just the input symbol number. The same procedure applies for output designation.

Example:

```
INPUT NODE IS > tfli     (Stands for element TF1 input)

OUTPUT NODE IS >a4i3     (Stands for adder A4 input #3.
                         Inputs for adders are numbered
                         clockwise from the adder output)
```

To temporarily alter the diagram prior to reduction the commands OPEN and CLOSE exist. These commands allow the lines connecting the various elements of the diagram to be temporarily opened or broken so that the reduction analysis can be done on a modification of the drawing.

The open command is used to designate which lines are to be opened. These points can be designated by name or by position using the crosshair. To activate the crosshair mode the key word CURSOR is entered in place of a alphanumeric location name.

Example:

```
   OPEN > tlo,a2i3,cursor     (Tl output, A2 third input,
                               and activate the cursor
                               mode.)
```

While the cursor is displayed on the screen, lines can be picked by entering a space when the crosshair is over the line to be opened. Entering an "E" ends the crosshair selection mode.

All lines that are opened will be designated by the letter "O" appearing on the line.

The close command is used to eliminate a previously created temporary opening. The syntax is similar to the open command.

Example:

```
   CLOSE > tlo,a2i3,cursor
```

When using the cursor to point to an opening to be closed the "O" marking the opening should be under the crosshair. Reversal of a temporary opening to the closed position is indicated by an "X" appearing over the "O".

Additionally, the key word ALL is available to close all of the temporary openings at one time.

Example:

```
CLOSE > all
```

To perform the actual reduction the command REDUCE NOW must be entered. The program will verify the state of the reduction switch to determin the type of reduction to perform, either online or EASY5. If a specialized EASY5 element has been used in the drawing then the state of the switch is ignored and EASY5 reduction is always used.

Example:

```
REDUCE NOW
ONLINE REDUCTION. CONFIRM Y/N > y
REDUCTION IN PROGRESS.
REDUCTION COMPLETE. ANSWER STORED IN TA1
```

Example:

```
REDUCE NOW
EASY5 REDUCTION. COMFIRM Y/N > y
FILE NAME OF ANALYSIS COMMANDS > filename
FILE NAME FOR
GENERATED EASY5 BATCH INPUT FILE > filename2
FILE CREATION IN PROGRESS.
BATCH INPUT FILE CREATED. NAME IS FILENAME2
```

The batch operation of EASY5 requires that the analysis steps desired also be input at reduction time (file "filename" in previous example). This is done by giving the name of a previously created file of commands or giving the key word INPUT. The word INPUT indicates that a file does not exist and that the information will be entered from the terminal in EASY5 card image format.

Save and Restore Mode

In order to allow interruption of a terminal session two commands, SAVE and RESTORE, are available. These commands allow the user to save his work from one terminal session to another. All aspects of the drawing topology and any numeric parameters defined are stored in a permanent file when requested. During restoration the drawing is copied to the CRT and the equation parameters restored to the database so that further block diagram definition can be continued. The current working location (CWL) for the diagram is not reestablished so the first DRAW command after a RESTORE must be a move. This reestablishes the CWL.

Example:

```
SAVE,filename1
Action: (Current data is stored in filename1.)
```

Example:

```
RESTORE,filename2
Action: (Data in filename2 becomes current
        data. This destroys any data in
        the database that was not stored
        by a previous SAVE command)
```

By using several different file names the user could store more than one block diagram for later reuse.

Control Flags

Certain features are controlled by flags. Each flag can be set by the user to be either on or off. the command format is flagname,on or flagname,off. These commands are legal at any time except in the DRAW mode.

Below is a list of the flags and the effects they have.

MINATURE

ON - produce minature drawing if block diagram is larger than drawing work space.

OFF - do not produce a minature drawing

REDUCTION

ON - diagram restricted to either continuous or single rate discrete system for interactive reduction.

OFF - multi rate and specialized blocks of EASY5 routine are allowed and the equivalent textual description will be generated.)

<u>Storage</u> <u>Variables</u>

Listed below are the names of the variables that the user can access. These names should be used to examine the contents of any polynominal. These names should also be used when setting a new input equal to a previously established value.

Name	Meaning
Tn	Transfer function for element Tn (includes both the numerator and demoninator polynominal)
TnN	Numerator for Tn
TnD	Denominator for Tn
TAn	Transfer function result from reduction of diagram.

III. Sample Operation

The following discussion and accompanying figures describe the actual steps needed to generate and reduce a small block diagram using the GRAPHIC system.

Program Initialization

When the program is activated, it prompts to determine what the user environment is for this session. The input identify the terminal communication rate and the terminal type. An example of this dialog is in Figure 3.

At this point the primary prompt of "COMMAND >" appears and either DRAW, PARAMETER, REDUCE, GROUP, SAVE, or RESTORE can be entered.

Diagram Definition

For this example, the DRAW mode will be used to create the diagram directly from the basic elements. The initial display in the drawing mode is shown in Figure 4.

Figures 5 through 22 depict the appearence of the display terminal as the diagram is defined. Table II lists the actual command codes entered and the figure that corresponds to the display status after that command is fully processed.

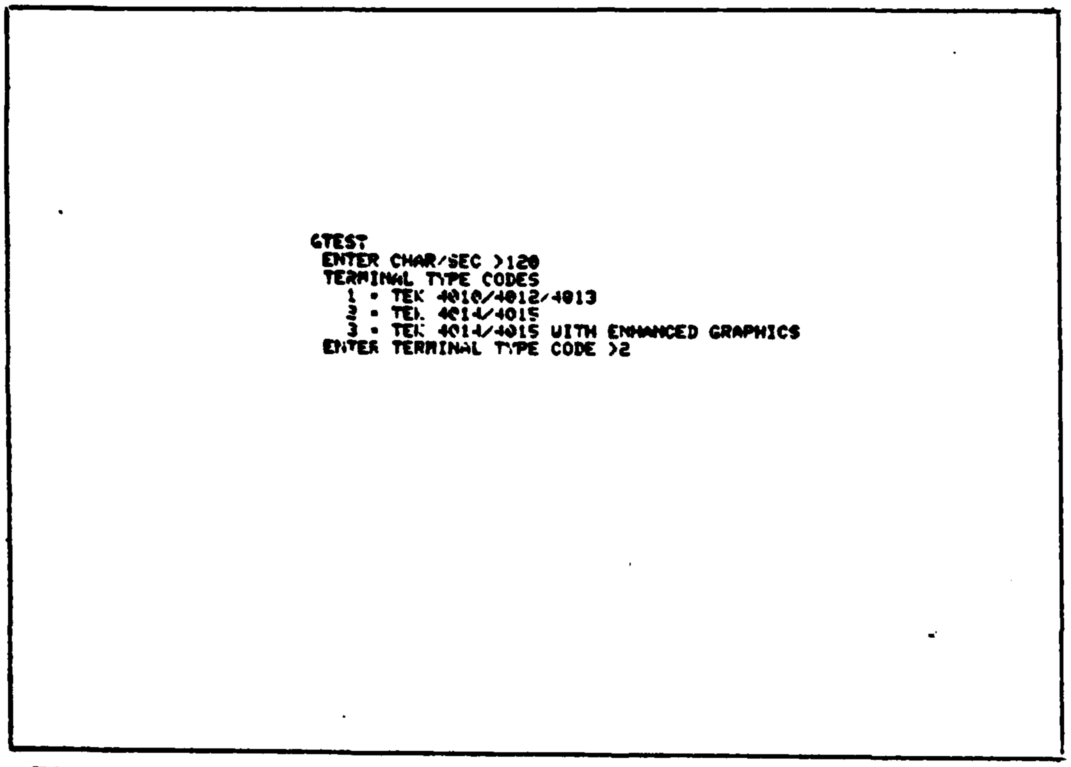
Parameter Definition

The PARAMETER mode is used next to define the values of the transfer functions that are specified in the drawing.

Figure 23 shows the complete PARAMETER mode display for the diagram in Figure 22.

Reduction

Once the diagram and its parameters have been completely defined, the system can be reduced through the REDUCE mode. This example shows only the EASY5 version of the reduction. Figure 24 shows the resulting file that is produced. The data shown in Figure 24 is the textual description of the diagram from Figure 22.



```
GTEST
 ENTER CHAR/SEC >120
 TERMINAL TYPE CODES
   1 = TEK 4010/4012/4013
   2 = TEK 4014/4015
   3 = TEK 4014/4015 WITH ENHANCED GRAPHICS
 ENTER TERMINAL TYPE CODE >2
```

Figure 3. Initialization Sequence

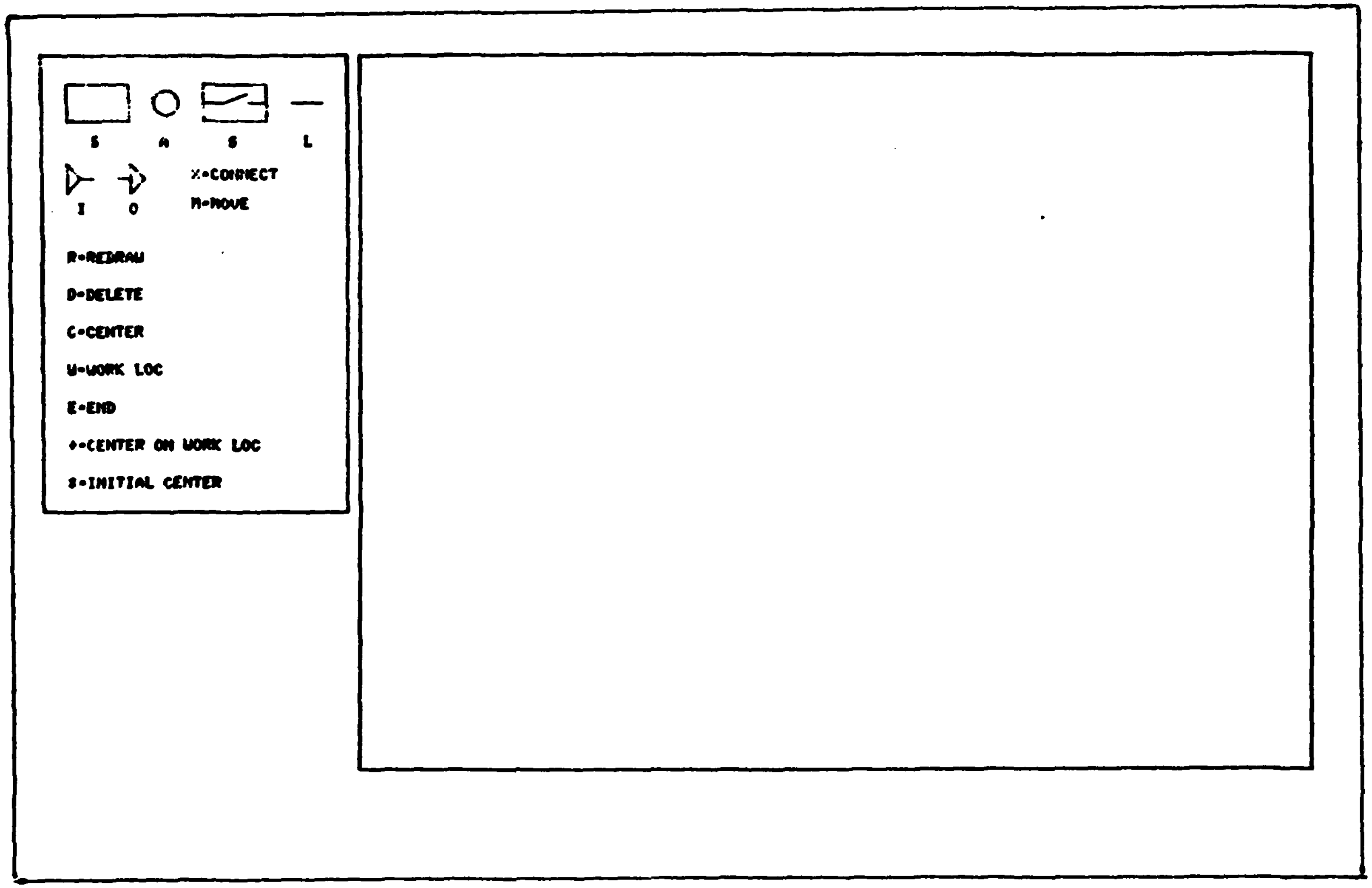


Figure 4. DRAW Mode Initial Display

Table II

Example Command Sequence

Step	Inputs	Figure
1	I	5
2	L	6
3	B	7
4	L	8
5	A	9
6	+	10
7	L	11
8	B	12
9	L	13
10	O	14
11	M,X	15
12	L	16
13	L	17
14	B	18
15	L	19
16	L	20
17	X	21
18	-	22

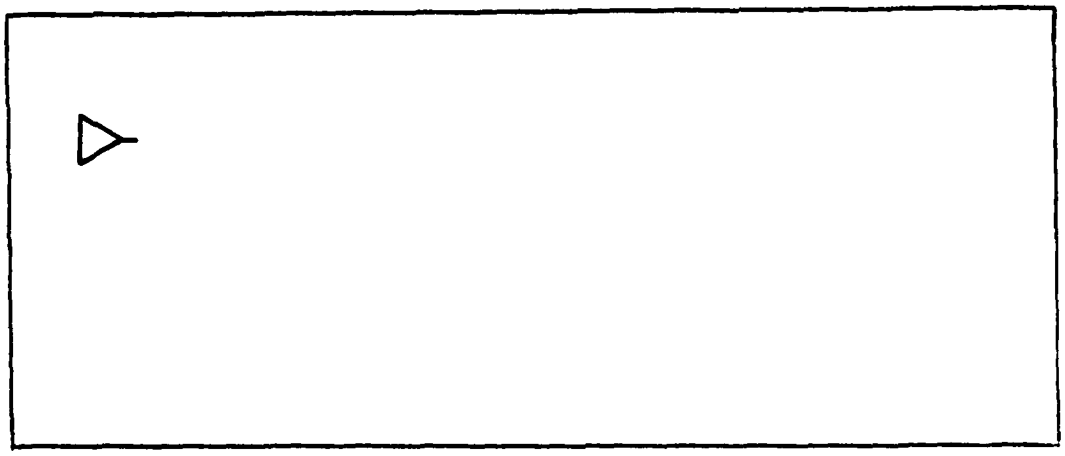


Figure 5. DRAW Mode Example Step 1

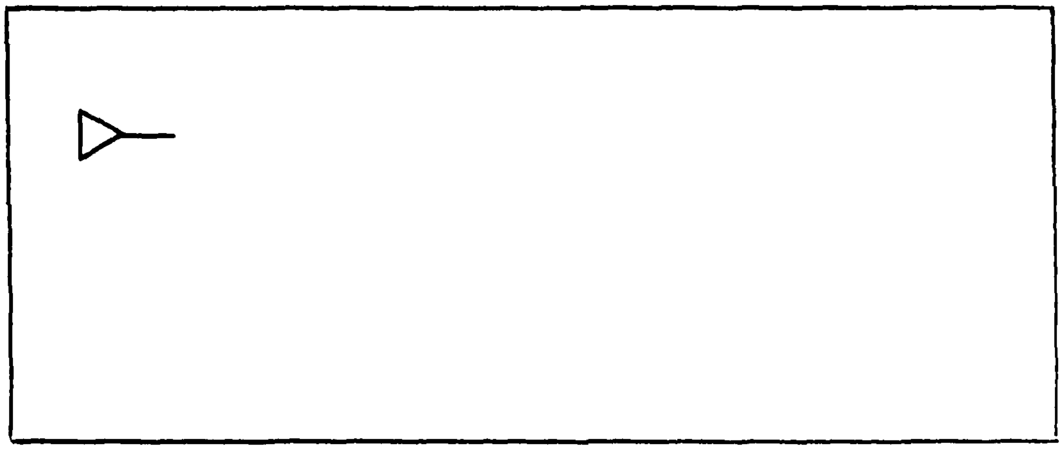


Figure 6. DRAW Mode Example Step 2

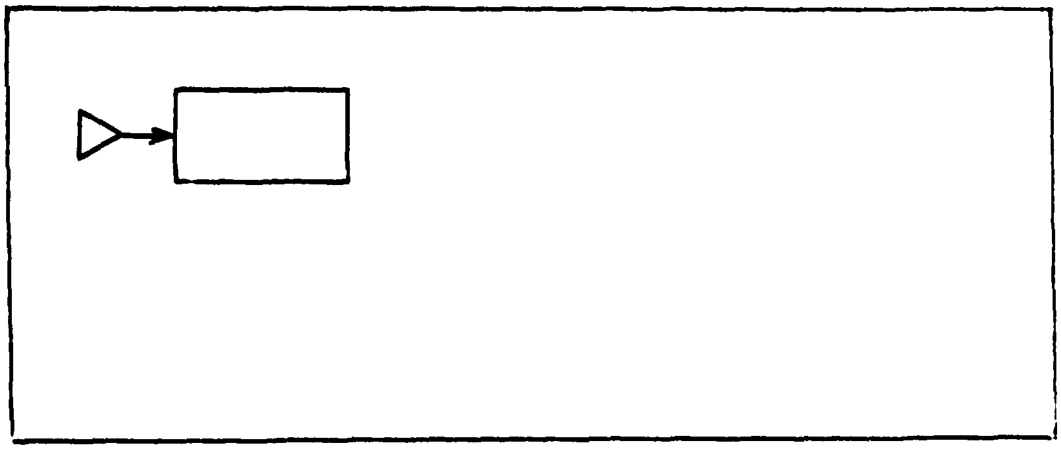


Figure 7. DRAW Mode Example Step 3

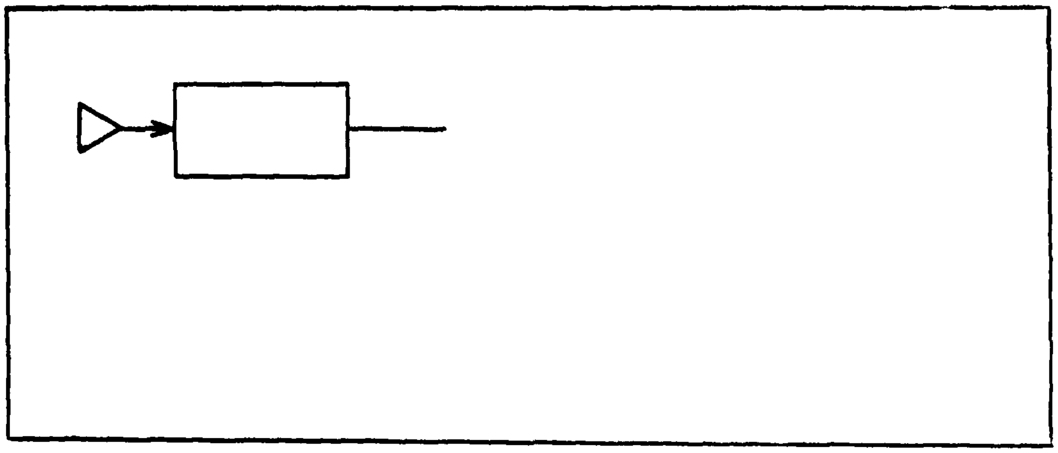


Figure 8. DRAW Mode Example Step 4

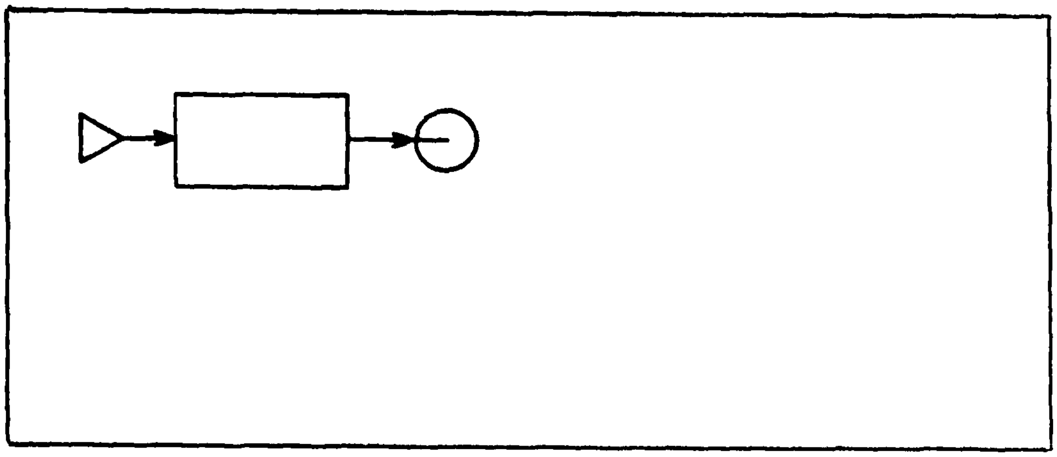


Figure 9. DRAW Mode Example Step 5

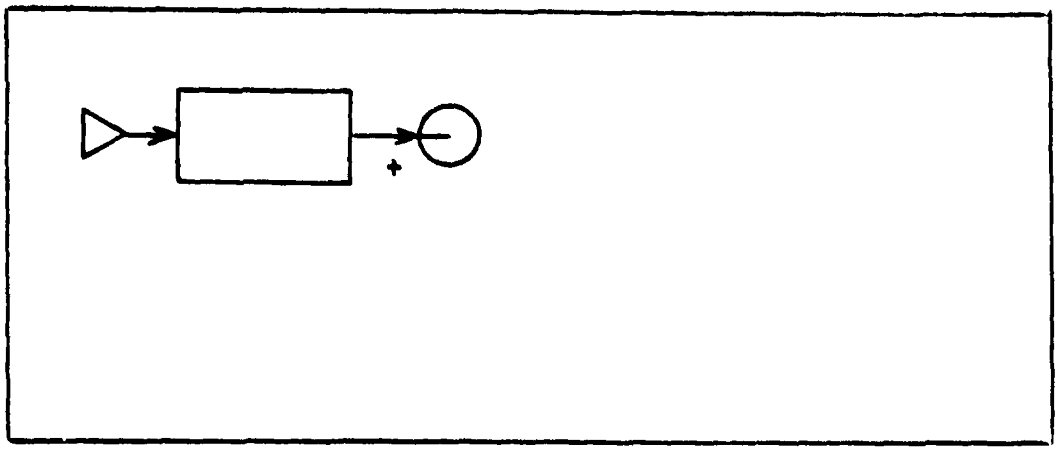


Figure 10. DRAW Mode Example Step 6

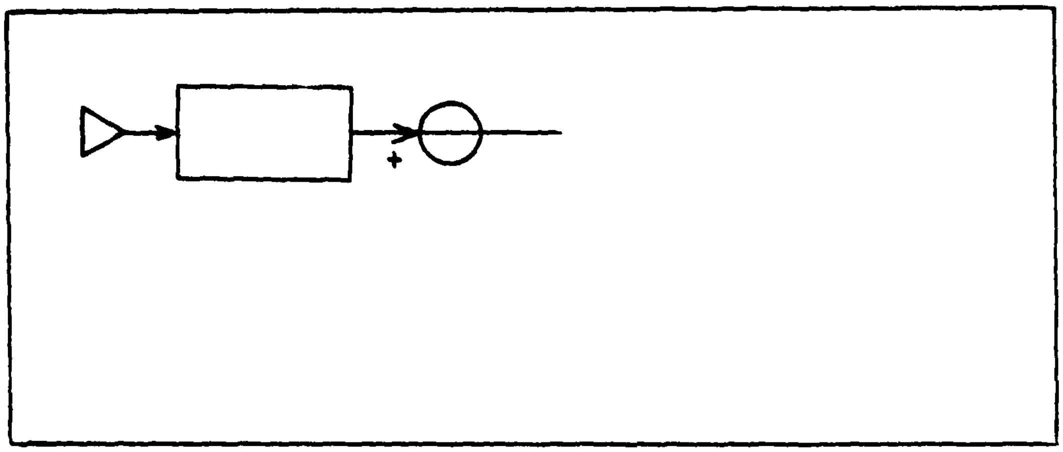


Figure 11. DRAW Mode Example Step 7

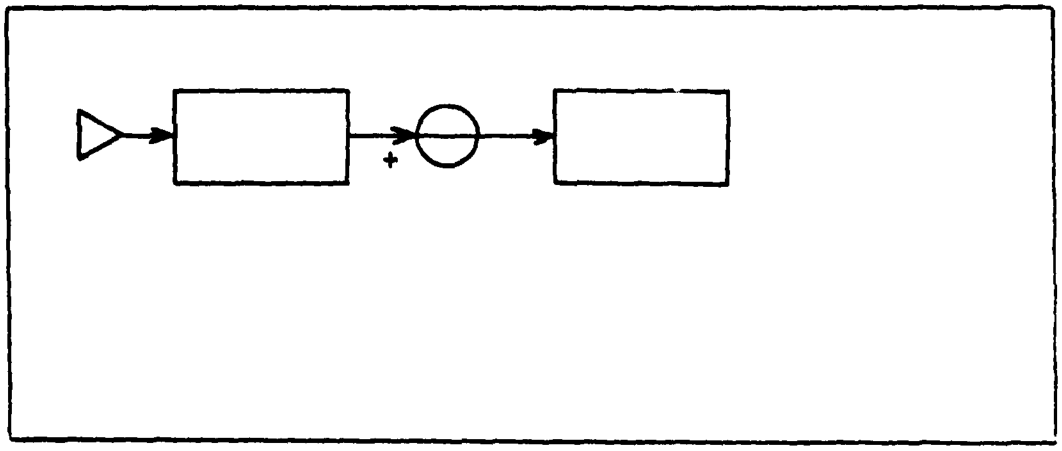


Figure 12. DRAW Mode Example Step 8

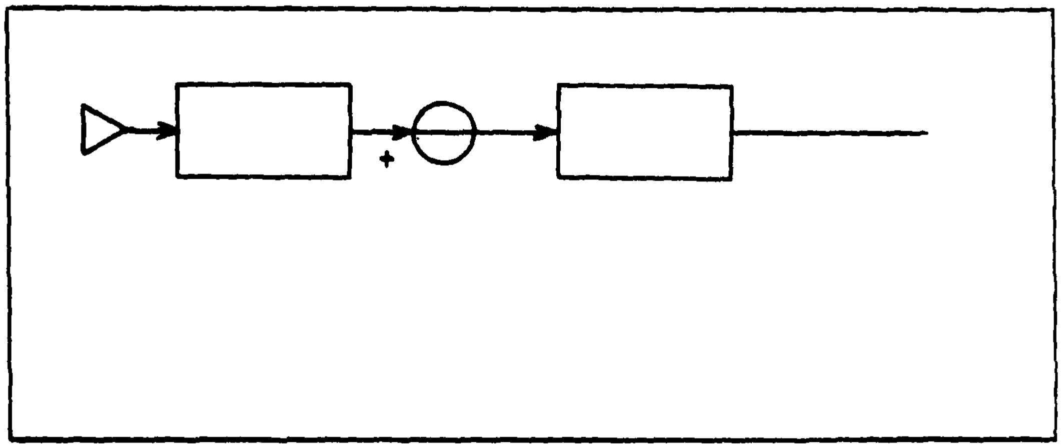


Figure 13. DRAW Mode Example Step 9

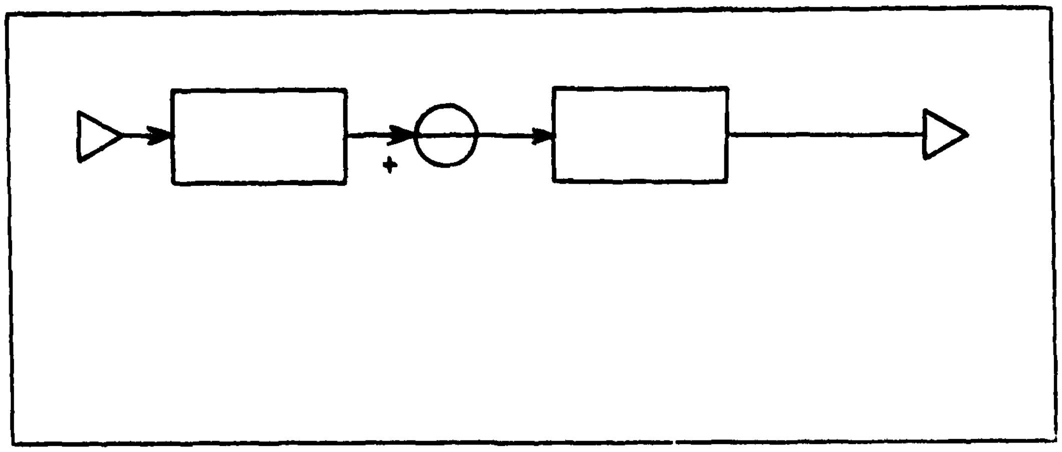


Figure 14. DRAW Mode Example Step 10

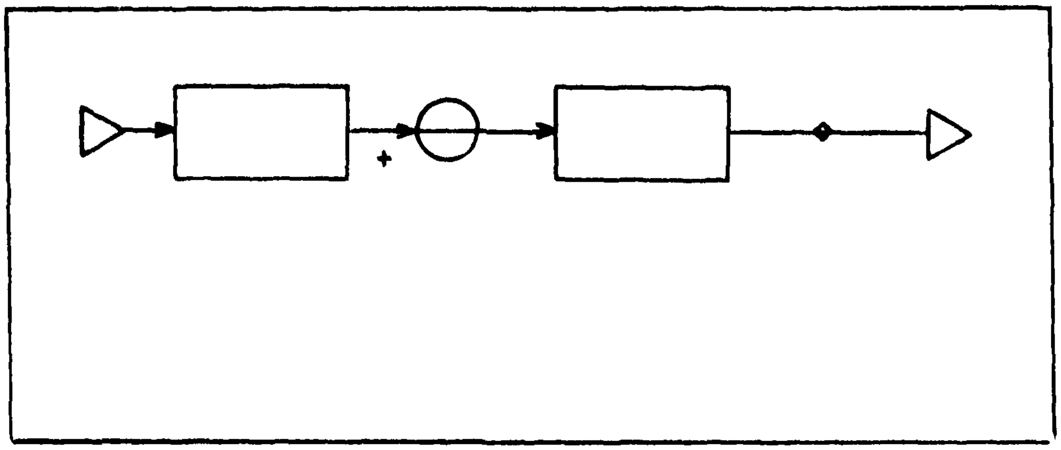


Figure 15. DRAW Mode Example Step 11

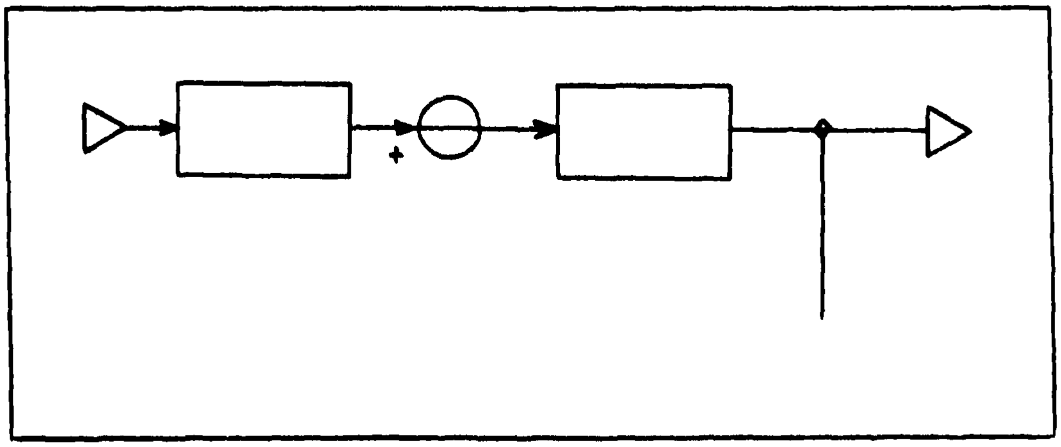


Figure 16. DRAW Mode Example Step 12

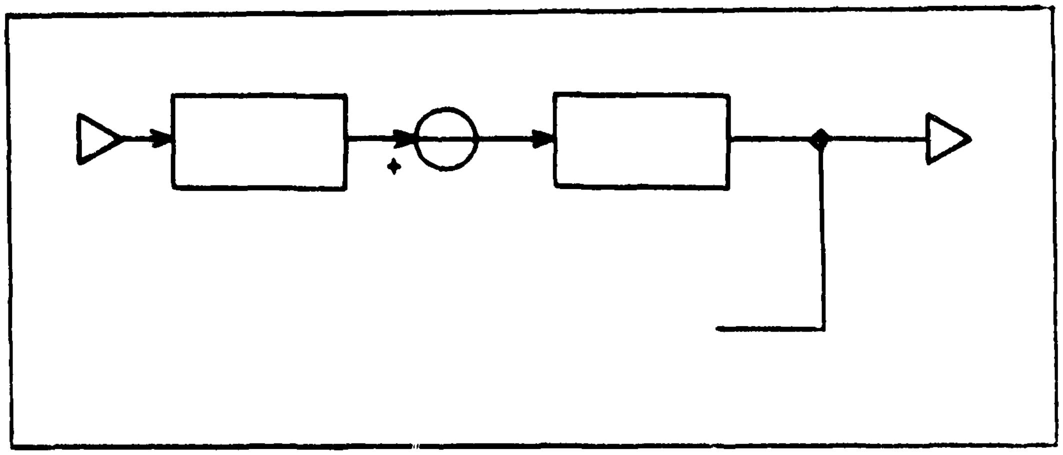


Figure 17. DRAW Mode Example Step 13

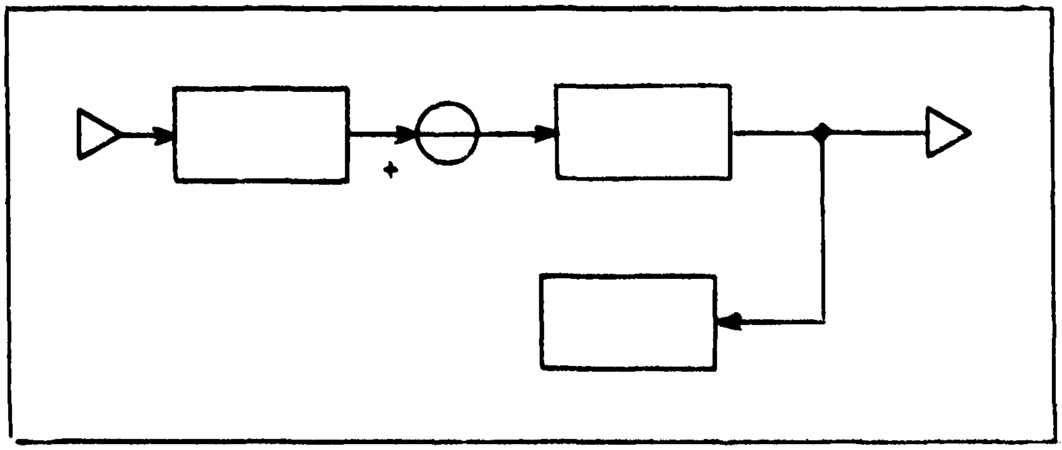


Figure 18. DRAW Mode Example Step 14

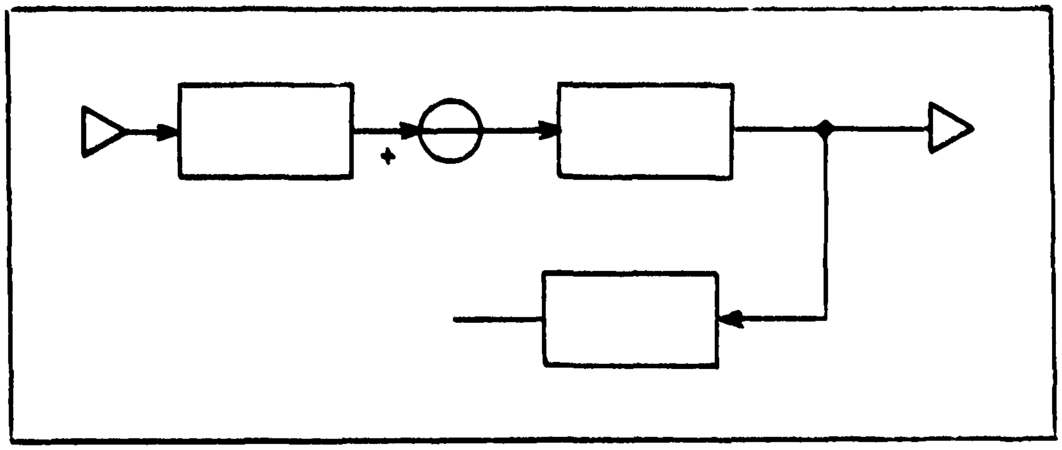


Figure 19. DRAW Mode Example Step 15

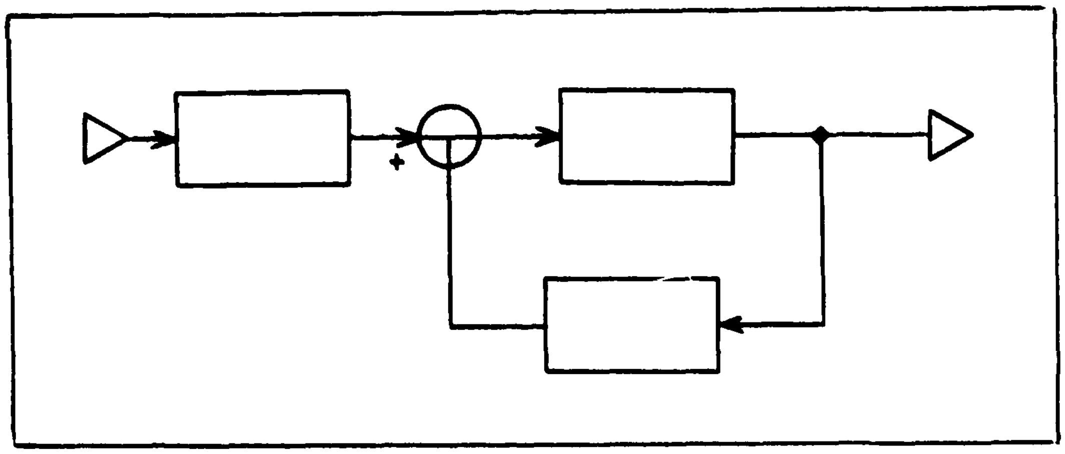


Figure 20. DRAW Mode Example Step 16

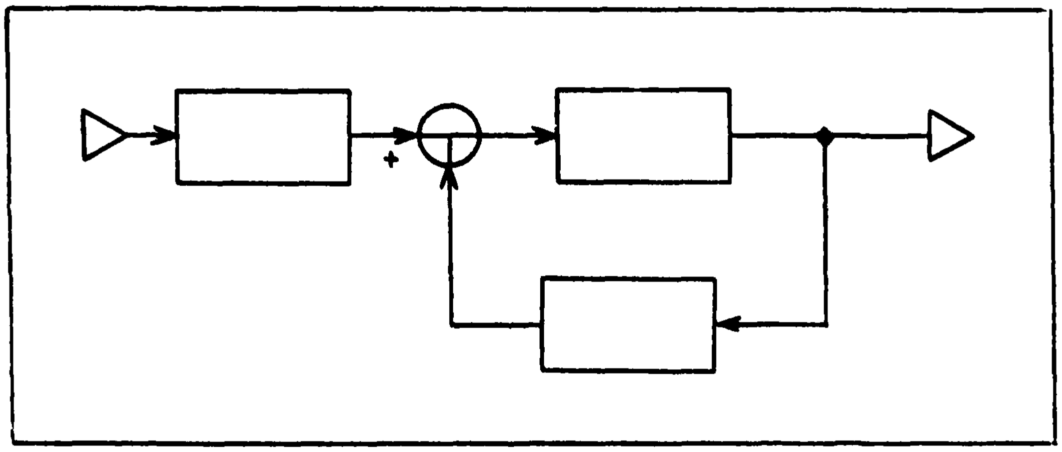


Figure 21. DRAW Mode Example Step 17

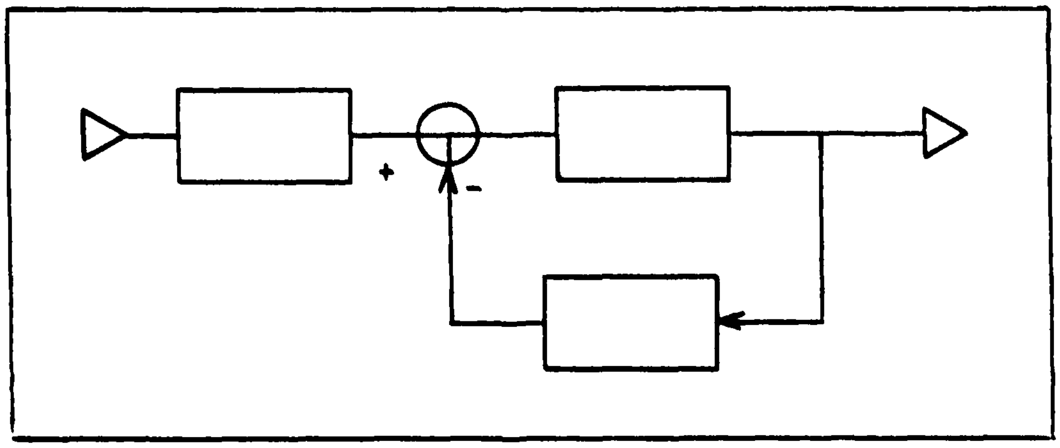
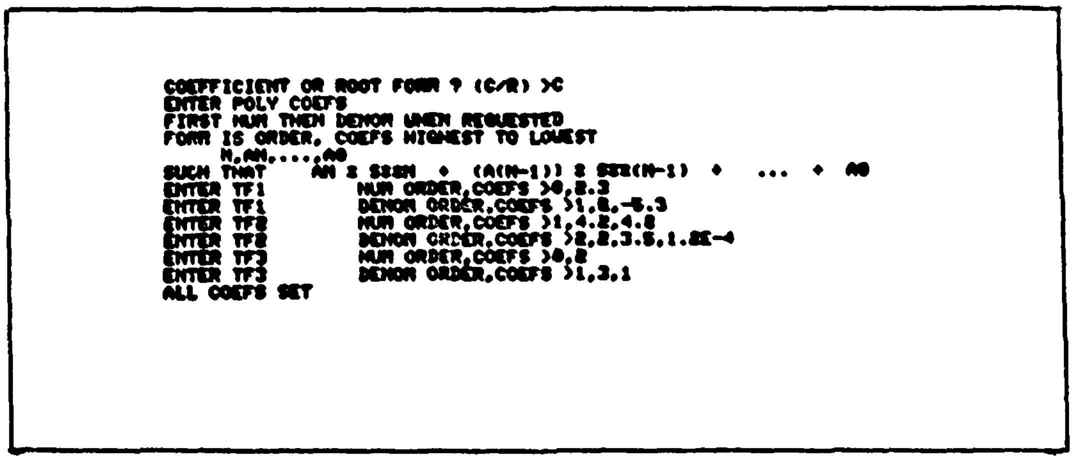
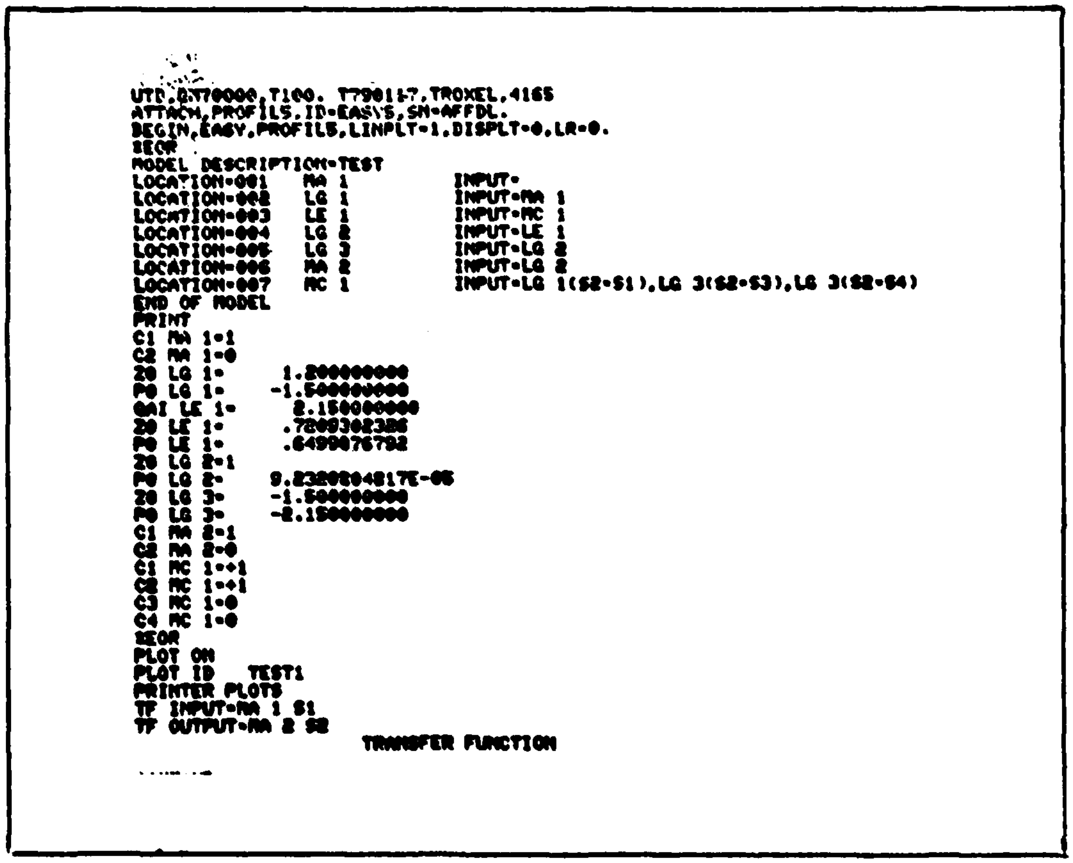


Figure 22. DRAW Mode Example Step 18



```
COEFFICIENT OR ROOT FORM ? (C/R) >C
ENTER POLY COEFS
FIRST NUM THEN DENOM WHEN REQUESTED
FORM IS ORDER, COEFS HIGHEST TO LOWEST
     N,AN,....,A0
SUCH THAT     AN * S**N  +  (A(N-1)) * S**(N-1)  +   ...  +  A0
ENTER TF1           NUM ORDER,COEFS >0,2.3
ENTER TF1           DENOM ORDER,COEFS >1,2,-5.3
ENTER TF2           NUM ORDER,COEFS >1,4.2,4.8
ENTER TF2           DENOM ORDER,COEFS >2,2,3.5,1.8E-4
ENTER TF3           NUM ORDER,COEFS >0,2
ENTER TF3           DENOM ORDER,COEFS >1,3,1
ALL COEFS SET
```

Figure 23. PARAMETER Mode Example



```
UTD,Q770000,T100. T790117,TROXEL,4165
ATTACH,PROFILS,ID=EASY5,SN=AFFDL.
BEGIN,EASY,PROFILS,LINPLT=1,DISPLT=0,LR=0.
EOR
MODEL DESCRIPTION=TEST
LOCATION=001    MA 1          INPUT=
LOCATION=002    LG 1          INPUT=MA 1
LOCATION=003    LE 1          INPUT=MC 1
LOCATION=004    LG 2          INPUT=LE 1
LOCATION=005    LG 3          INPUT=LG 2
LOCATION=006    MA 2          INPUT=LG 2
LOCATION=007    MC 1          INPUT=LG 1(S2=S1),LG 3(S2=S3),LG 3(S2=S4)
END OF MODEL
PRINT
C1 MA 1=1
C2 MA 1=0
Z0 LG 1=     1.200000000
P0 LG 1=    -1.500000000
GAI LE 1=     2.150000000
Z0 LE 1=     .7809302386
P0 LE 1=     .6499076792
Z0 LG 2=1
P0 LG 2=    9.2320804817E-06
Z0 LG 3=    -1.500000000
P0 LG 3=    -2.150000000
C1 MA 2=1
C2 MA 2=0
C1 MC 1=+1
C2 MC 1=+1
C3 MC 1=0
C4 MC 1=0
EOR
PLOT ON
PLOT ID   TEST1
PRINTER PLOTS
TF INPUT=MA 1 S1
TF OUTPUT=MA 2 S2
                         TRANSFER FUNCTION
```

Figure 24. Example of EASY5 Format Reduction

IV. Current Implementation Status

The version of GRAPHIC that is currently implemented allows a subset of the total GRAPHIC system to be used. The exact nature of the variations from the features described in this manual are addresed on a mode by mode basis below.

DRAW Mode

In the DRAW mode all features except the GROUP command are implemented. Command CONNECT has a code of X instead of C. Command CENTER has codes of C,+,# instead of +,@,# respectively (see Appendix B). The BOX command forces all box elements to be the arbitrary TF type.

GROUP Mode

None of the GROUP mode features are implemented.

PARAMETER Mode

In the PARAMETER mode only the automatic sequence mode is available. Setting sampling periods is not allowed. Use of the calculator feature and varible names are not allowed. All parameters for transfer functions are considered as Laplace, S domain, functions. The PARAMETER mode is activated by the command "E" in response to the prompt "COMMAND >".

REDUCE Mode

The REDUCE mode provides only EASY5 form reduction. The OPEN and CLOSE features for temporary alteration of the diagram topology are not available. All reductions must be from an input to an output specified by giving their respective numbers. The REDUCE mode is activated by entering "T" after the prompt "COMMAND >".

SAVE Mode

The SAVE mode catalogs only the drawing database and only if it was assigned to a permanent file device.

RESTORE Mode

The RESTORE mode restores only the drawing database.

Summary

The current implementation of GRAPHIC was produced to verify the proposed design documented in this manual. This implementation allows generation and translation of any block diagram representing a time continuous linear control system. The output file produced by the program can be used directly as an input to the EASY5 program.

APPENDIX B

UPDATES TO USER'S MANUAL

Contents

I. Introduction

As a result of the user test conducted to verify the design proposed for the GRAPHIC system, certain changes were made to the design proposed in Appendix A. This appendix documents those changes.

Since the actual changes made to the design affect only two of the fourteen commands in the graphic input mode, only the changed material is documented here.

Section II details the new features that were designed into the CENTER command and Section III details the changes that were made to the CONNECT command.

II. Center Command Design

Center Command - Code:+,@,#. The CENTER command shifts the existing drawing around in the work area to create more space for defining additional elements. There are three forms of the CENTER command, one associated with each of the command codes; +, @, and #.

One form of the CENTER operation is to take the location designated by the crosshair and redraw the diagram with that location centered in the work area. The code for this is +. Any element or point within the work area or miniature area may be pointed to and when the + is entered the drawing will be erased, shifted as required, and redrawn.

The other two forms of the CENTER command operate with default locations. Entering @ causes the current working location (CWL) to be centered in the drawing when the diagram is redrawn. Entering # causes the diagram to be centered as it was initially when the draw mode was first activated.

III. Connect Command Design

Connect Command - Code: C. The connect command is used to specify that at the current crosshair location a connection is to be formed. There are three uses of the CONNECT command all controlled by the same command code. The difference is in the graphic result produced on the terminal screen.

A connection can be used to split an output path so that one element can feed two other elements. This is accomplished by pointing the crosshair at a line and entering the code C. A small connecton diamond will appear over the point where the output splits and a line can be drawn away from the diamond through the use of the LINE command.

The second use of the connection command is to join the current path with an already existing path. This joining can be to an adder or to the input side of any individual element that does not already have its input in use. To form the connection correctly the current path must end in a line that is drawn to the proper spot on the existing target element. For adders the proper spot is the center of the element, but for all other elements it is the input side of the element. When the code C is entered after the line is drawn a arrowhead will be generated automatically to indicate the signal flow direction. Additionally, if the connection was to form an adder input, the sign of the input will be requested as described in the ADDER command.

The last use of the CONNECT command is to continue drawing a previously interrupted path. This is accomplished by positioning the crosshair over the output side of the last element in the interrupted path and entering the code C.

If the requested connection con not be performed, the terminal will beep to indicate an error and a message will be posted in the text area. An example of an error that could occur is that no element is found to connect with when the command is issued.

Appendix C

CORE LIBRARY

CORE LIBRARY

The CORE library is a collection of subroutines designed to allow the creation of three dimensional drawings. It is a partial implementation of the ACM SIGGRAPH proposed standard for device independent graphics support.

This particular implementation was developed as a MA 6.68 class project at the Air Force Institute of Technology in which the author participated. The version used with the GRAPHIC system was further modified by the author.

The library consists of application program callable subroutines that allow drawing in three dimensional space. Any viewpoint can be established and either parallel or perspective views can be produced.

Description of User Callable Routines.

The following routines comprise the user callable subroutines in the CORE library. Those routines that are defined in the proposed SIGGRAPH standard have the corresponding SIGGRAPH paragraph number given in square breckets, [Ref. #].

CORSYS3 (delta-x, delta-y, delta-z, scale-x, scale-y,
scale-z, rotation-x, rotation-y, rotation-z)

CORSYS3 specifies the orientation of the users coordinate system relative to the world coordinate system. The transform is applied as scale, rotate, then translate. The rotations are positive conterclockwise (as viewed from a point on the positive axis looking toward the origin) and applied in the order z-axis, y-axis, and then x-axis.

VUREF (x, ~~y, z) [Ref.~~ 5~~.2.1.1]~~

VUREF specifies the reference point in the world system for basing the viewing system. Eye point (see PERSPEC), view plane distance (see VUDIST), view depth (see VUDEPTH), and viewing window (see WINDOW) are measured relative to this point.

VUNORM (delta-x, delta-y, delta-z) [Ref. 5.2.1.2]

VUNORM specifies the view direction in the world system.

VUDIST (distance) [Ref. 5.2.1.3]

VUDIST specifies the projection or view plane distance from the reference point, measured in world units in the direction of the normal (see VUNORM).

VIEWUP3 (delta-x, delta-y, delta-z) [Ref. 5.2.1.6]

VIEWUP3 specifies (from its projection into the view plane) which direction is up in the view plane.

WINDOW (left-edge, right-edge, bottom-edge, top-edge) [Ref. 5.2.2.2]

WINDOW specifies the location of the window measured in the view plane relative to the projection of the reference point down the view normal.

VUDEPTH (front-distance, back-distance) [Ref. 5.2.1.8]

VUDEPTH specifies the front and back clipping planes. The distance is measured from the reference point in the direction of the view normal.

VUPORT (left-edge, right-edge, bottom-edge, top-edge) [Ref. 5.2.1.10]

VUPORT specifies the location of the display area on the plotting or display device. Allowable range for the parameters is 0.0 to 1.0 (normalized device space).

PARALL (delta-x, delta-y, delta-z) [Ref. 5.2.1.4]

PARALL specifies that the projection type is to be parallel, with the objects projected along the given vector into the view plane.

PERSPEC (offset-x, offset-y, offset-z) [Ref. 5.2.1.5]

PERSPEC specifies that the projection type is to be perspective with the eye point offset by the given values from the reference point. The look direction is in the direction given by the view normal.

MOVE3 (x, y, z) [Ref. 2.2.1]

MOVE3 specifies that the current beam location is to be moved to the given user cooridnates.

DRAW3 (x, y, z) [Ref. 2.2.2.1]

DRAW3 specifies that a line is to be drawn from the current beam location to the user cooridnate location given.

STARTG [Ref. 7.2.1.1]

STARTG specifies that the graphic system is to be activated and all required variables initialized.

ENDG [Ref. 7.2.1.2]

ENDG specifies the termination of operations under the graphic system.

OUTLINE

OUTLINE specifies that a outline around the existing viewport is to be drawn.

<u>Utitilization</u> <u>Notes</u>.

The system will detect and request changes to values when they would result in mathematically undefined operations. When such a condition occurrs all subsequent calls to MOVE3 and DRAW3 are ignored until the viewing parameters are changed.

The combination of calls to WINDOW and VUDEPTH establish a view volume. Only objects appearing inside the view volume will be displayed. The volume is a parallelepiped if PARALL was called and a truncated pyramid if PERSPEC was called.

Geometrically meaningless combinations of the parameters can result depending on the users parameters. Example: window right edge less than window left edge or front distance greater than back distance. The result is that all objects are clipped and not displayed since there can be no view volume with the given parameters. The system will never adjust any user parameter except for the front distance given in VUDEPTH. The front distance will be adjusted when perspective projections are in use to insure that the front clipping plane is in front of the eye point. This adjustment will be done in the following manner: "if the front distance measured from the view reference point in the direction of the view normal is less than the the eye distance measured in the same manner, then the front distance will be reset automatically by the system to be equal to the eye distance plus one tenth (eyedistance + .1)."

Consistancy of the system parameters supplied by the user are checked before performing the first MOVE3 of DRAW3 after a viewing parameter was changed. This approach to the parameter checking allows the individual parameters to be changed in any order and does not require that they be consistant until a MOVE3 or DRAW3 is requested.

All coordinate systems, user, world, and view, are left handed systems. All subroutine parameters are real values.

To assist in debug the routine STATPRT is available to print a list of the viewing parameters in effect at any given moment. The routine will print on the output file a list of all the user setable parameters and also some of the internal tramsformation matrices derived from the user parameters. These internal paramaters will have correct values only if either a MOVE3 or a DRAW3 has been executed since the last change to any viewing parameter. The user specifiable parameters are shown correctly at all times.

APPENDIX D

DESCRIPTION OF GRAPHIC PROGRAM STRUCTURE

Contents

I. Introduction

This appendix documents the design of the current implementation of the GRAPHIC program. It describes all the routines that comprise the GRAPHIC software.

The GRAPHIC program is written in FORTRAN for the Control Data Corporation (CDC) CYBER 175 series computer. It is designed to be operated interactively from a Tektronix 4000 series storage tube graphics terminal.

In addition to the source code described in this appendix the GRAPHIC program makes use of several library packages. These library routines provide services that are not available in the normal FORTRAN environment. Specifically, the TEKLIB, NOSLIB, and CORE libraries are used (Ref. 6,18,2,19). The TEKLIB library provides routines for communicating with the Tektronix terminal. The NOSLIB library provides routines for managing permanent files. The CORE library provides routines for three dimensional drawing. Additional details on the CORE library are discussed in Appendix C.

II. Detailed Routine Descriptions

This section discusses the general features of each routine, program and function on an individual basis. The exact nature of the function that each routine performs is documented in detail in the actual source code. The list of modules is divided into two groups, those modules that are entry points for overlays and those modules that are not entry points. The descriptions are in alphabetical order by module within each group.

Overlay Entry Point Modules

The relationships between the mode names, overlay numbers, and actual source names for the programs are shown in Table I.

DRAW Program. The DRAW program operates as a master control switch, sequencing the operation of all commands allowed during the DRAW mode. Whenever the graphic crosshair is present on the screen the DRAW program is waiting to act on input data from its service routines to execute the requested command.

When DRAW is first entered it draws the initial display on the screen by calling DRAWINT. It then calls UPDATE to open and initialize the graphic database. If the graphic database contained data, the diagram defined by that data is drawn on the screen through a call to REDRAW.

At this point DRAW begins the loop that forms a command filter allowing any of the DRAW mode commands The routine

Table I

Overlay Names

Operation	Overlay	Source Name
(command)	0,0	TEST
DRAW	1,0	DRAW
SAVE	2,0	SAVE
RESTORE	3,0	RESTORE
PARAMETER	4,0	EQUATE
REDUCE	5,0	REDUCE

LOCATOR is used to obtain the command code and crosshair location. ARROW is called to insure that an arrowhead is automatically produced if one is needed. Then depending on the code received, one of the following routines is called: USERGRP, LINE, BOX, MOVEPT, ADDER, SAMPLE, CONNECT, CENTER, REDRAW, IN, OUT, DELETE, or WORKLOC. When the code "E" for END is received the loop is terminated, UPDATE is called to insure that all data has been written to the graphic database, and control is returned to the main overlay.

EQUATE Program. The EQUATE program provides all of the process control for the PARAMETER mode operations. Its only specific function is to select the correct routine to collect the numeric parameters. If coefficient mode input is requested the EQCOEFS routine is called. If root mode input is requested the EQROOTS routine is called.

REDUCE Program. The REDUCE program sequences the operation of routines during REDUCE mode operation. Routine EZ is called to perform the entire reduction to EASY5 format. REDUCE posts the success or failure message on the terminal indicating the result of the attempted reduction.

RESTORE Program. The RESTORE program allows reuse of graphic databases that have been previously saved via the SAVE mode. Specifically, it uses the routines in the NOSLIB to attach a permanent file as file DRAWIN. The permanent file selected is specified by the input that REDUCE receives from the terminal.

SAVE Program. The SAVE program catalogs the existing graphic database to allow it to be reused at a later time. This can be done only if the file DRAWIN was assigned to a permanent file device prior to activating GRAPHIC.

TEST Program. The TEST program forms the main overlay of the GRAPHIC system. It contains the common data storage areas that are shared by more than one overlay. When activated, the program initializes the common data areas. Once the initialization is complete, the program then loops between getting mode requests from the user and activating overlays to execute the requested mode.

Other Modules

All other modules in the GRAPHIC system are subroutines or functions. The paragraphs that follow summarize the functions of each routine so as to form a guide to the source listings.

Subroutine ABSTWDS. The ABSTWDS subroutine converts XY location values from the integer Device Coordinate System (DCS) to the real User Coordinate System (UCS) for storage in the graphic database. The actual conversion depends on the current mapping between the DCS and UCS systems.

Subroutine ADDER. The ADDER subroutine handles creation of an adder in response to the command code A.

Subroutine ALERT. The ALERT subroutine closes and saves the graphic database mass storage file if a fatal error occurs during graphic operation.

Subroutine ARROW. The ARROW subroutine handles the creation of arrowheads to indicate the direction of signal flow. This routine decides whether an arrowhead is required or not based on the previous command and the current command.

Subroutine ASIGNS. The ASIGNS subroutine displays the arrowheads and input signs for the second and third inputs to adders. It is used only during redraw operations.

Subroutine BLANK. The BLANK subroutine is called to erase the display screen and reset the scrolling messages to start in the upper left hand corner of the text area.

Subroutine BRKLINE. The BRKLINE subroutine is called by CLINK in response to a connect command that causes a line to be split. This routine makes the calls to UPDATE to remove the old line and replace it with two shorter lines.

Subroutine BOX. The BOX subroutine handles the creation of a transform box symbol in response to a command code B.

Subroutine CENTER. The CENTER subroutine handles repositioning the viewing window to display a different part of the diagram in response to a command code of C.

Subroutine CHECK. The CHECK subroutine rotates the coordinate system to provide a better check to see if a diagonal line element is the item designated by a connect command.

Subrotuine CLINK. The CLINK subroutine handles updating the topology links in the graphic database for all connect commands that were preceded by a move command.

Subroutine CLINKIN. The CLINKIN subroutine handles updating the topology links in the graphic database for all connect commands that were not preceded by a move command.

Subroutine CMDHELP. The CMDHELP subroutine posts an explanation of the modes available in GRAPHIC.

Subroutine CMDMODE. The CMDMODE subroutine is the command filter that sorts the response to the prompt "COMMAND >" and determines which overlay to activate.

Subroutine CONNECT. The CONNECT subroutine handles the request for a connection and verifys that it was accomplished successfully.

Subroutine CONSYM. The CONSYM subroutine is called when a connection diamond is needed. It creates the symbol on the display and stores a record of it in the graphic database.

Subroutine DADDER. The DADDER subroutine controls the drawing of the adder symbol. It contains the vector drawing commands that control the shape of the symbol.

Subroutine DARROW. The DARROW subroutine controls the drawing of the arrow symbols. It contains the vector drawing commands that control the shape of the symbol.

Subroutine DBOX. The DBOX subroutine controls the drawing of the box symbol. It contains the vector drawing commands that control the shape of the symbol.

Subroutine DCONECT. The DCONECT subroutine controls the drawing of the connection symbol. It contains the vector drawing commands that control the shape of the symbol.

Subroutine DELDATA. The DELDATA subroutine removes elements from the graphic database in response to delete commands.

Subroutine DELETE. The DELETE subroutine manages all operations necessary to accomplish the deletion of an element. This routine manages the search for, deletion of, and marking of the element designated by the crosshair.

Subroutine DIN. The DIN subroutine controls the drawing of the input symbol. It contains the vector drawing commands that control the shape of the symbol.

Subroutine DLIMIT. The DLIMIT subroutine maintains the records on the maximum size of the current diagram being defined. This information is needed to control the automatic generation of the miniature drawing.

Subroutine DOUT. The DOUT subroutine controls the drawing of the output symbol. It contains the vector drawing commands that control the shape of the symbol.

Subroutine DRAWINT. The DRAWINT subroutine is responsible for establishing the viewport boundaries on the display screen and drawing outlines around them. The routine also sets the mapping between the User Coordinate System (UCS) and the display face.

Subroutine DSAMPLER. The DSAMPLER subroutine controls the drawing of the sampler symbol. It contains the vector drawing commands that control the shape of the symbol.

Subroutine ELEMNUM. The ELEMNUM subroutine determins the next available number for each symbol type in the diagram. This number is used to generate the symbol labals that appear on the display screen.

Subroutine ERRPRT. The ERRPRT subroutine is used to print debug error messages on the display screen.

Subroutine EQCOEFS. The EQCOEFS subroutine handles the user interface to collect and store transfer function definitions when the user has requested the coefficient mode of input.

Subroutine EQROOTS. The EQROOTS subroutine handles the user interface to collect and store transfer function definitions when the user has requested the root mode of input.

Subroutine EZ. The EZ subroutine handles the sequencing of the routines that create the textual output file that can serve as input to the EASY5 program.

Subroutine EZANAL. The EZANAL subroutine creates the analysis section of the commands in the EASY5 text file. This routine gets input output pairs from the user and then transcribes this into an EASY5 format transfer function calculation request.

Subroutine EZBLOCK. The EZBLOCK subroutine controls the translation of block diagrams into the textual format required by EASY5. It activates the FLOW secondary overlay to create the signal flow database. After this, it activates the EZWRITE secondary overlay to produce the textual model description.

Subroutine EZHEAD. The EZHEAD subroutine creates the control card section of the EASY5 text file.

Subroutine FATAL. The FATAL subroutine is designed to force a fatal error. This routine is used during debug to test the ALERT subroutine.

Subroutine FORMLAB. The FORMLAB subroutine forms the visible labels used to idenitify the elements in the block diagram. It combines the integer element number and the ASCII element type into an alphanumeric ASCII label.

Subroutine GETSIGN. The GETSIGN subroutine queries the user for the sign of each adder input as the input is created.

Subroutine GPROMPT. The GPROMPT subrotine controls the contents of the DRAW mode prompt area. This routine draws the figures and text that is displayed there.

Function IEZELMN. The IEZELMN function creates EASY5 element numbers. It accepts an integer and returns the two character ASCII equivalent.

Function ILETTER. The ILETTER function converts values into ASCII. It accepts the integer ASCII code and returns the character that is the equivalent code. This function is used to support the Tektronix interface.

Subroutine IN. The IN subroutine handles creation of an input element in response to the command code I.

Subroutine INITIAL. The INITIAL subroutine initializes all values in the main overlay. It also initializes the graphics support package.

Function INUMBER. The INUMBER function accepts ASCII characters and converts them to their equivalent numeric code.

Subroutine LINE. The LINE subroutine handles creation of an line in response to the command code L.

Subroutine LISTDEL. The LISTDEL subroutine handles deleting items from the linked list index after the item has been removed from the graphic database in response to a delete request.

Subroutine LOCATOR. The LOCATOR subroutine determines where the crosshair was pointing during the last input. The value returned is the location relative the User Coordinate System used for the graphic database.

Function LOCEZFM. The LOCEZFM function accepts integer location numbers and converts them into their three character EASY5 equivalent form.

Subroutine LUPDATE. The LUPDATE subroutine maintains the linked list that forms the index into the entries of the graphic database. As elements are drawn, this routine adds pointers to the linked list.

Subroutine MDRAW. The MDRAW subroutine determins if a miniature drawing should be produced. If one is needed, this routine sets the coordinate mapping from the database to the display screen.

Subroutine MESSAGE. The MESSAGE subroutine is used to scroll all messages that are output in the text area.

Subroutine MINDRAW. The MINDRAW subroutine produces the actual miniature drawing of the diagram when requested by MDRAW.

Subroutine MOVEPT. The MOVEPT subroutine moves the current working location (CWL) to the position specified by the crosshair.

Function NUMVAL. The NUMVAL function returns the numeric value of an element label. This value is used in storing the element's parameters in the numeric database.

Subroutine OUT. The OUT subroutine handles creation of an output element in response to the command code O.

Subroutine PUTSIGN. The PUTSIGN subroutine posts the sign of the first input to an adder symbol on the display screen.

Subroutine REDRAW. The REDRAW subroutine scans the graphic database and draws the current version of the diagram on the display screen.

Subroutine SAMPLE. The SAMPLE subroutine handles creation of an sampler in response to the command code S.

Subroutine SEARCH. The SEARCH subroutine supports the delete command. It searches the graphic database and returns a list of elements that are near the current crosshair location.

Subroutine SMALL. The SMALL subroutine sets the character size for the Tektronix 4014/15 terminals to the smallest available size.

Subroutine SMALWIN. The SMALWIN subroutine draws the corner marks in the miniature area to mark what part of the entire diagram is visible in the work area.

Subroutine TEKPRT. The TEKPRT subroutine is used to print the labels that appear on the elements within the work area.

Subroutine TERMINL. The TREMINL subroutine produces the main prompt "COMMAND >" and reads the users response.

Subroutine TEXTSET. The TEXTSET subroutine puts the terminal in text mode with the cursor positioned at any desired location.

Subroutine TFDATAB. The TFDATAB subroutine handles all access to the numeric database for reading and writing transfer function data.

Subroutine UPDATE. The UPDATE subroutine handles all additions to graphic database in response to DRAW mode commands given by the user.

Subroutine WRSTATE. The WRSTATE subroutine reads and writes the linked list index of elements and the drawing size information that is stored as specialized records in the graphic databaase.

Subroutine WORKLOC. The WORKLOC subroutine displays the square text cursor at the position of the current working location (CWL).

Subroutine ZERO. The ZERO subroutine is used to initialize the buffer where information on the current element being formed in to be stored.

APPENDIX E

ABBREVIATIONS AND MNEMONICS

ABBREVIATIONS AND MNEMONICS

Term	Definition
ACM	Association for Computing Machinery.
AFFDL/FGC	Air Force Flight Dynamics Laboratory / Control Dynamics Branch.
CDC	Control Data Corporation.
CENTER	GRAPHIC DRAW mode command that moves the window over the block diagram to allow display of different sections of the diagram.
CONNECT	GRAPHIC DRAW mode command that causes two independent signal flow paths to be connected.
CORE	Name of the software library that supports three dimensional graphics.
CRT	Display tube that forms the face of the Tektronix terminal.
CWL	Current Working Location. The location in the block diagram that is the output of the most recently produced element symbol.
CYBER	Model of computer manufactured by the Control Data Corporation.
DCS	Device Coordinate System. The coordinate system used to control drawing on the display face.
DRAW	Mode of operation in GRAPHIC that allows the drawing of a block diagram to be defined.
EASY5	A control system analysis program developed by Boeing Aerospace Company.

Term	Definition
FLOW	Software routine in GRAPHIC that creates a signal flow graph from the graphical block diagram definition.
FORTRAN	Name of the programming language that GRAPHIC is written in.
GRAPHIC	Name given to the software system being designed in this paper.
GROUP	Mode of operation in GRAPHIC that allows generation of collections of elements to form block diagrams of subsystems.
INTERCOM	Name of the interactive timesharing system used to run the GRAPHIC program.
MOVE	GRAPHIC DRAW mode command that allows repositioning of the current drawing location (CWL).
NOSLIB	Name of the software library that supports manipulation of permanent files.
NOS/BE	Computer operating system that the GRAPHIC program is compatible with.
PARAMETER	Mode of operation in GRAPHIC that allows definition of the numeric parameters for transfer functions and sampling periods.
REDUCE	Mode of operation in GRAPHIC that allows the block diagram to be reduced to an equivalent transfer function or an equivalent textual description.
RESTORE	Mode of operation in GRAPHIC that allows restoration of previous work from a stored file.
SAVE	Mode of operation in GRAPHIC that allows all work to be saved in a file for reuse.
SIGGRAPH	Special Interest Group for Graphics, a group within the ACM.

Term	Definition
TEKLIB	Name of the software library that supports interfaces to the Tektronix 4000 series terminals.
TOTAL	A control system analysis program.
UCS	User Coordinate System. The arbitrary coordinate system that the block diagram is referenced to.
UPDATE	Software routine within GRAPHIC that alters the graphic database.

Vita

Donald E. Troxel was born on December 8, 1950 in Washington, DC, the son of David I. Troxel and Jean M. Troxel. In 1973 he graduated from Lehigh University in Bethlehem, Pennsylvania with a Bachelor of Science in Electrical Engineering and was commissioned as a Distinguished Graduate of the Air Force ROTC program. He entered active duty upon commissioning and served in Air Force Systems Command at Hanscom AFB, Massachusetts until 1978, when he entered the Air Force Institute of Technology.

Permanent mailing address:

Capt. Donald E. Troxel
5242 Garfield Ave.
Pennsauken, NJ 08109

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/MA/79D-1	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) GRAPHICAL INPUT METHODOLOGY FOR COMPUTER AIDED ANALYSIS OF CONTROL SYSTEMS		5. TYPE OF REPORT & PERIOD COVERED MS Thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Donald E. Troxel Capt USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Flight Dynamics Laboratory (AFFDL/FGC) Wright-Patterson AFB, Ohio 45433		12. REPORT DATE December 1979
		13. NUMBER OF PAGES 166
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

Joseph P. Hipps, Captain, USAF
Director, Public Affairs

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Graphics
Interactive Graphics
Control Systems
Block Diagrams
Human Interfaces

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the design of a graphical input language and CAD system for specifying control system block diagrams. The goal to develop a graphical means of defining block diagrams that avoided the need to create the textual descriptions required by most existing traditional analysis packages was accomplished. Generalized requirements for a successful human interface were developed. A graphical input methodology was developed that meets these human constraints. The CAD system designed allows graphical definition of linear systems and translation of these systems into forms acceptable by traditional analysis programs.